DOCUMENT RESUME

ED 079 991 EM 011 366

AUTHOR Friend, Jamesine
TITLE Computer-Assisted Instruction in Programming: A Curriculum Description. Technical Report Number 211.
INSTITUTION Stanford Univ., Calif. Inst. for Mathematical Studies in Social Science.
SPONS AGENCY Office of Naval Research, Washington, D.C. Personnel and Training Research Programs Office.
REPORT NO IMSSS-TR-211
PUB DATE 31 Jul 73
NOTE 93p.

EDRS PRICE MF-$0.65 HC-$3.29
DESCRIPTORS *Community Colleges; *Computer Assisted Instruction; *Computer Science Education; Course Descriptions; Individualized Instruction; Junior Colleges; On Line Systems; Program Descriptions; *Programing; Time Sharing
IDENTIFIERS AID; Algebraic Interpretive Dialogue

ABSTRACT

A course has been developed which provides an introduction to computer programing for community college students who have taken high school algebra. Instruction is presented by computer and a student reference manual is provided. The content resembles that of other introductory courses in computer programing, including topics on stored programs, use of variables, input-output control, syntax of algebraic and logical expressions, use of functions and subroutines, conditional clauses, and branching techniques. The instructional system implements, under computer control, teaching strategies useful to human tutors, such as individualizing the content, pace, and sequence of instruction, allowing for sufficient student control, tailoring wrong answer messages, and providing remedial and extra-credit work. Students are required to interact on-line with a commercially prepared editor-interpreter similar to those found in many timesharing environments. Performance data from the instructional program and the editor-interpreter are stored for retrieval and analysis. The organization of the course, types of exercises used, and content of each lesson are documented and an appendix lists the concepts associated with each exercise. (Author/PB)

FILMED FROM BEST AVAILABLE COPY

ED 079991

COMPUTER-ASSISTED INSTRUCTION IN PROGRAMMING: A CURRICULUM DESCRIPTION

EM

BY

JAMESINE FRIEND

TECHNICAL REPORT NO. 211

JULY 31, 1973

PSYCHOLOGY AND EDUCATION SERIES

EM 011 366

INSTITUTE FOR MATHEMATICAL STUDIES IN THE SOCIAL SCIENCES
STANFORD UNIVERSITY
STANFORD, CALIFORNIA

FILMED FROM BEST AVAILABLE COPY

TECHNICAL REPORTS

PSYCHOLOGY SERIES

INSTITUTE FOR MATHEMATICAL STUDIES IN THE SOCIAL SCIENCES

(Place of publication shown in parentheses; if published title is different from title of Technical Report, this is also shown in parentheses.)

(For reports no. 1 - 44, see Technical Report no. 135.)

50 R. C. Atkinson and R. C. Calfee. Mathematical learning theory. January 2, 1963. (In B. B. Wolman (Ed.), Scientific Psychology. New York: Basic Books, Inc., 1965. Pp. 254-275)

51 P. Suppes, E. Crothers, and R. Weir. Application of mathematical learning theory and linguistic analysis to vowel phoneme matching in Russian words. December 28, 1962.

52 R. C. Atkinson, R. Calfee, G. Sommer, W. Jeffrey and R. Shoemaker. A test of three models for stimulus compounding with children. January 29, 1963. (J. exp. Psychol., 1964, 67, 52-58)

53 E. Crothers. General Markov models for learning with inter-trial forgetting. April 8, 1963.

54 J. L. Myers and R. C. Atkinson. Choice behavior and reward structure. May 29, 1963. (Journal math. Psychol., 1964, 1, 170-203)

55 R. E. Robinson. A set-theoretical approach to empirical meaningfulness of measurement statements. June 10, 1963.

56 E. Crothers, R. Weir and P. Palmer. The role of transcription in the learning of the orthographic representations of Russian sounds. June 17,. 1963.

57 P. Suppes. Problems of optimization in learning a list of simple items. July 22, 1963. (In Maynard W. Shelly, II and Glenn L. Bryan (Eds.), Human Judgments and Optimality. New York: Wiley. 1964. Pp. 116-126)

58 R. C. Atkinson and E. J. Crothers. Theoretical note: all-or-none learning and intertrial forgetting. July 24, 1963.

59 R. C. Calfee. Long-term behavior of rats under probabilistic reinforcement schedules. October 1, 1963.

60 R. C. Atkinson and E. J. Crothers. Tests of acquisition and retention, axioms for paired-associate learning. October 25, 1963. (A comparison of paired-associate learning models having different acquisition and retention axioms, J. math. Psychol., 1964, 1, 285-315)

61 W. J. McGill and J. Gibbon. The general-gamma distribution and reaction times. November 20, 1963. (J. math. Psychol., 1965, 2, 1-18)

62 M. F. Norman. Incremental learning on random trials. December 9, 1963. (J. math. Psychol., 1964, 1, 336-351)

63 P. Suppes. The development of mathematical concepts in children. February 25, 1964. (On the behavioral foundations of mathematical concepts. Monographs of the Society for Research in Child Development, 1965, 30, 60-96)

64 P. Suppes. Mathematical concept formation in children. April 10, 1964. (Amer. Psychologist, 1966, 21, 139-150)

65 R. C. Calfee, R. C. Atkinson, and T. Shelton, Jr. Mathematical models for verbal learning. August 21, 1964. (In N. Wiener and J. P. Schoda (Eds.), Cybernetics of the Nervous System: Progress in Brain Research. Amsterdam, The Netherlands: Elsevier Publishing Co., 1965. Pp. 333-349)

66 L. Keller, M. Cole, C. J. Burke, and W. K. Estes. Paired associate learning with differential rewards. August 20, 1964. (Reward and information values of trial outcomes in paired associate learning. (Psychol. Monogr., 1965, 79, 1-21)

67 M. F. Norman. A probabilistic model for free-responding. December 14, 1964.

68 W. K. Estes and H. A. Taylor. Visual detection in relation to display size and redundancy of critical elements. January 25, 1965, Revised 7-1-65. (Perception and Psychophysics, 1966, 1, 9-16)

69 P. Suppes and J. Donio. Foundations of stimulus-sampling theory for continuous-time processes. February 9, 1965. (J. math. Psychol., 1967, 4, 202-225)

70 R. C. Atkinson and R. A. Kinchla. A learning model for forced-choice detection experiments. February 10, 1965. (Br. J. math stat. Psychol., 1965, 18, 184-206)

71 E. J. Crothers. Presentation orders for items from different categories. March 10, 1965.

72 P. Suppes, G. Groen, and M. Schlag-Rey. Some models for response latency in paired-associates learning. May 5, 1965. (J. math. Psychol., 1966, 3, 99-128)

73 M. V. Levine. The generalization function in the probability learning experiment. June 3, 1965.

74 D. Hansen and T. S. Rodgers. An exploration of psycholinguistic units in initial reading. July 6, 1965.

75 B. C. Arnold. A correlated urn-scheme for a continuum of responses. July 20, 1965.

76 C. Izawa and W. K. Estes. Reinforcement-test sequences in paired-associate learning. August 1, 1965. (Psychol. Reports, 1966, 18, 879-919)

77 S. L. Blehart. Pattern discrimination learning with Rhesus monkeys. September 1, 1965. (Psychol. Reports, 1966, 19, 311-324)

78 J. L. Phillips and R. C. Atkinson. The effects of display size on short-term memory. August 31, 1965.

79 R. C. Atkinson and R. M. Shiffrin. Mathematical models for memory and learning. September 20, 1965.

80 P. Suppes. The psychological foundations of mathematics. October 25, 1965. (Colloques Internationaux du Centre National de la Recherche Scientifique. Editions du Centre National de la Recherche Scientifique. Paris: 1967. Pp. 213-242)

81 P. Suppes. Computer-assisted instruction in the schools: potentialities, problems, prospects. October 29, 1965.

82 R. A. Kinchla, J. Townsend, J. Yellott, Jr., and R. C. Atkinson. Influence of correlated visual cues on auditory signal detection. November 2, 1965. (Perception and Psychophysics, 1966, 1, 67-73)

83 P. Suppes, M. Jerman, and G. Groen. Arithmetic drills and review on a computer-based teletype. November 5, 1965. (Arithmetic Teacher, April 1966, 303-309.

84 P. Suppes and L. Hyman. Concept learning with non-verbal geometrical stimuli. November 15, 1965.

85 P. Holland. A variation on the minimum chi-square test. (J. math. Psychol., 1967, 3, 377-413).

86 P. Suppes. Accelerated program in elementary-school mathematics -- the second year. November 22, 1965. (Psychology in the Schools, 1966, 3, 294-307)

87 P. Lorenzen and F. Binford. Logic as a dialogical game. November 29, 1965.

88 L. Keller, W. J. Thomson, J. R. Tweedy, and R. C. Atkinson. The effects of reinforcement interval on the acquisition of paired-associate responses. December 10, 1965. (J. exp. Psychol., 1967, 73, 268-277)

89 J. I. Yellott, Jr. Some effects on noncontingent success in human probability learning. December 15, 1965.

90 P. Suppes and G. Groen. Some counting models for first-grade performance data on simple addition facts. January 14, 1966. (In J. M. Scandura (Ed.), Research in Mathematics Education. Washington, D. C.: NCTM, 1967. Pp. 35-43.

91 P. Suppes. Information processing and choice behavior. January 31, 1966.

92 G. Groen and R. C. Atkinson. Models for optimizing the learning process. February 11, 1966. (Psychol. Bulletin, 1966, 66, 309-320)

93 R. C. Atkinson and D. Hansen. Computer-assisted instruction in initial reading: Stanford project. March 17, 1966. (Reading Research Quarterly, 1966, 2, 5-25)

94 P. Suppes. Probabilistic inference and the concept of total evidence. March 23, 1966. (In J. Hintikka and P. Suppes (Eds.), Aspects of Inductive Logic. Amsterdam: North-Holland Publishing Co., 1966. Pp. 49-65.

95 P. Suppes. The axiomatic method in high-school mathematics. April 12, 1966. (The Role of Axiomatics and Problem Solving in Mathematics. The Conference Board of the Mathematical Sciences, Washington, D. C. Ginn and Co., 1966. Pp. 69-76.

(Continued on inside back cover)

U S DEPARTMENT OF HEALTH
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION
THIS DOCUMENT HAS BEEN REPRO
DUCED EXACTLY AS RECEIVED FROM
THE PERSON OR ORGANIZATION ORIGIN
ATING IT. POINTS OF VIEW OR OPINIONS
STATED DO NOT NECESSARILY REPRE
SENT OFFICIAL NATIONAL INSTITUTE OF
EDUCATION POSITION OR POLICY

ED 079991

COMPUTER-ASSISTED INSTRUCTION IN PROGRAMMING:

A CURRICULUM DESCRIPTION

by

Jamesine Friend

TECHNICAL REPORT NO. 211

July 31, 1973

PSYCHOLOGY & EDUCATION SERIES

Reproduction in Whole or in Part is Permitted for Any
Purpose of the United States Government

This research was sponsored by the Personnel and Training Research Programs, Psychological Sciences Division, Office of Naval Research, under Contract No. N00014-67-A-0112-0054, Contract Authority Identification Number, NR 154-326.

INSTITUTE FOR MATHEMATICAL STUDIES IN THE SOCIAL SCIENCES
STANFORD UNIVERSITY
STANFORD, CALIFORNIA

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Institute for Mathematical Studies in the Social Sciences - Stanford University Stanford, California 94305	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

Computer-Assisted Instruction in Programming: A Curriculum Description

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical Report

5. AUTHOR(S) (First name, middle initial, last name)

Jamesine Friend

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
31 July 1973	80	5

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-67-A-0012-0054 b. PROJECT NO. NR 154-326 c. d.	Technical Report No. 211 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Personnel & Training Research Programs, Office of Naval Research Arlington, VA 22217

13. ABSTRACT

The course provides an introduction to computer programming for community college students who have taken high school algebra, and it is equivalent to a three quarter-unit course in computer science. All instruction is presented by computer, and a supplementary student manual is provided for reference. The course content resembles that of other introductory courses in computer programming and includes the topics of stored programs, use of variables, input and output control, syntax of algebraic and logical expressions, use of functions and subroutines, conditional clauses, and branching techniques. The instructional system implements under computer control teaching strategies that might be used by a human tutor such as individualizing the content, pace, and sequence of instruction, allowing for sufficient student control, tailoring wrong answer messages, and providing both remedial and extra-credit work. Students are required to interact 'on-line' with a commercially prepared editor-interpreter similar to those found in many timesharing environments. Performance data from both the instructional program and the editor-interpreter are automatically stored for later retrieval and analysis. The organization of the course, types of exercises used, and content of each lesson are documented and an appendix lists the concepts associated with each exercise in the course.

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Algebraic Interpretive Dialogue (AID) Computer-assisted Instruction (CAI) Computer Programming Computer Science Education						

COMPUTER-ASSISTED INSTRUCTION IN PROGRAMMING:

A CURRICULUM DESCRIPTION[1]

Jamesine Friend
Stanford University

In 1967 the Institute for Mathematical Studies in the Social Sciences began the development of a computer controlled course in computer programming. This course is an introduction to computer programming for community college students who have had high school algebra, and it is equivalent to a three unit, one quarter community college course. All instruction is presented by computer, so that the course can be used where there are no qualified instructors of programming; a supplementary student manual used as a reference book is provided. In the first stage of development about one-third of the course was implemented on the PDP-1 computer and tested on a small group of volunteers, mostly Stanford University students. The course was then revised, completed, and implemented on the larger PDP-10 computer. Since then several hundred students in several schools and colleges have been enrolled for the course. In 1970 data collection routines were added to the instructional system so that the course could be used for research.

Students taking the course communicate with the PDP-10 computer at Stanford University using KSR Model 33 teletypewriters which are connected to the computer by ordinary telephone lines. A teletypewriter may be located anywhere that electricity and telephone lines are available--in an office, a classroom, or even a private home; in practice, these student terminals are usually put into small classrooms in clusters of

four to eight machines. Each terminal communicates independently with the computer, and each student works independently of other students. A student may work at any hour (provided the computer is in operation at that time) and at any terminal. The communication between the computer and the student takes the form of a "conversation" in which the computer and student take turns typing on the terminal, the computer presenting instruction and problems and the student replying by typing his answers; the computer then analyzes the student's answers, tells the student whether he is right or wrong, and supplies further instruction.

A standard Model 33 KSR teletypewriter is ordinarily used although several similar models could also be employed. These machines have limitations that affect both the style and content of the material being taught. For example, the Model 33 teletypewriter is a slow output device, delivering only ten characters per second. Since a college student can read much faster than ten characters per second, large quantities of text are printed by teletypewriter only at the risk of boring the student. Partly for this reason, instruction in the course is given in small, succinct paragraphs, never more than several hundred words, and usually less than one hundred words. Another limitation of the Model 33 teletypewriter is its inability to produce graphic characters. Depending upon the subject to be taught this limitation is more or less severe; elementary geometry, for example, is essentially impossible, but social studies are less difficult. For a course in computer programming the limitation is not severe, although flow charting, which ordinarily forms a part of an introductory course in programming, is best omitted. It is possible, using the limited set of characters available, to produce

fairly acceptable flow charts, but this is so time-consuming, both in the original programming of the display and in the printing itself, that the inclusion of flow charts is of questionable value.

Other than the omission of flow charts, the content of the course is quite similar to other introductory courses in programming. The course teaches the concept of stored programs, the use of variables, and introduction to input and output, the syntax of algebraic expressions, definitions of functions, conditional clauses and the syntax of logical statements, conditional and unconditional branching, core and disk storage, and an introduction to subroutines. Some of these subjects, such as disk storage and subroutines, are discussed only briefly, while others, such as the syntax of algebraic expressions and the use of conditional branches, are covered more extensively.

The language that is being taught in this course is AID (Algebraic Interpretive Dialogue), an algebraic language in the same class as the programming languages ALGOL, FORTRAN, and BASIC. AID is in some respects superior to these other languages as a first programming language. Since it is interpreted rather than compiled, students can use direct commands that will be executed immediately, giving them an opportunity for early hands-on experience; in a language that is compiled the user cannot execute a command until his program, written in the proper format, is stored, and compiled, and the run command is given. Another advantage of AID is that it is a subset of English extended by the language of elementary mathematics, making AID commands and programs easy to read. For example, these typical commands can be read and understood with no instruction:[2]

```
SET X = 5
SET Y = 1/X
TYPE Y IF X + Y < X/2
```

One disadvantage of AID is that it is less well-known and not as available as BASIC, ALGOL, FORTRAN, and a number of other languages. This means that a student who learns AID is less likely to be able to put his knowledge to immediate use. In partial compensation for this, AID is sufficiently similar to the more widely used algebraic languages that a student who knows AID well can readily learn one of these other languages with very little instruction. Also, several variants of AID bearing different names are implemented on a variety of computers. One of these, the original on which AID was modeled, is JOSS, a language designed at RAND Corporation for use by engineers and scientists. Another nearly identical language is FOCAL, which is implemented for several Digital Equipment Corporation computers.

When AID was chosen as the language to teach, the choice had rapidly narrowed down to AID and BASIC. BASIC, an algebraic language designed at Dartmouth College as a beginner's language, is widely known and used, and has several very useful advanced commands for matrix manipulation that are lacking in AID. BASIC is an excellent beginner's language, and perhaps would have been the choice for this course except for two factors: (1) AID as an interpreted language was felt to be more responsive, and (2) (a more pragmatic consideration), BASIC was not at that time implemented on the IMSSS system. Soon after development of the AID course began, a BASIC compiler became available for the PDP-10, and a BASIC course was subsequently written at Stanford. The BASIC course

was patterned after the AID course and uses the same instructional system.

This instructional system was designed to give tutorial instruction under computer control and implements several teaching strategies that might be used by a human tutor. Tutorial instruction, as contrasted with the lecture method of teaching, requires an ability to tailor the sequence and content of the instruction to the individual student. In order to do this the student's desires and abilities must be taken into account. The analysis of an individual student's ability depends upon prior accumulation and analyses of his responses to exercises and problems, and the bulk of the instructional system consists of routines for analyzing such responses. Some of these routines, such as the routine for analyzing a response to a true-false exercise, are quite simple; others are relatively complex, allowing for such possibilities as "correct but incomplete" and "partially correct," and for a wide range of equivalent answers. All of the analysis routines used in the system depend upon the student's response being in an expected form. Thus, a student confronted with an arithmetic problem like 50^2 is expected to reply with a numeral like 2500 or 2500.0 or 2.5×10^3; responses like "twenty-five hundred" or "The answer is 2500" are not recognized as correct answers and the instructional program will reply "No. Type a number." Because the student's response must be phrased in a restricted form, each exercise is written so that the expected form of the response is clear to the student. Despite the limitation of restricted forms for student responses, a great variety of problem types can be used, and the correctness of responses determined with precision, permitting considerable individualization of instruction.

Individualization of instruction is also achieved in the course by varying both the amount of instruction given and the sequence of instruction. Both of these are largely under student control. Students control the amount of instruction in each exercise by a device called the "hint" option. A student may request additional information for any exercise by asking for a "hint." For very difficult problems, several hints may be provided while for simpler exercises, such as true-false exercises, no hints are provided. If a student requests a hint for one of these easier exercises, he is simply told that there are no hints available. For most exercises students are allowed an unlimited number of opportunities to respond. A student may attempt to answer, then ask for a hint, try again, ask for another hint, etc. If a student tries one problem several times and feels that he is making no headway, he may ask for the answer and continue with the lesson; in this way each student is allowed to judge for himself how much time he should profitably spend on each problem.

The sequence of instruction is also controlled by the student. There is an automatic sequence of exercises that each student takes, unless he initiates a change of sequence. This sequence may differ for different students depending upon their ability. A student may interrupt the instructional program at any time to request a different exercise. He will then proceed in the automatic sequence from the exercise he requested until he again interrupts the program. Students use this feature either to skip over lessons or portions of lessons, or to review some topic they have previously studied.

Although the students may override the automatic sequencing of exercises provided by the program, most prefer to follow the prescribed sequence more or less as given. Even if students choose to take exercises and lessons exactly as given, they will not all receive an identical set of exercises, since the program provides alternative branches based on current assessments of student ability. Such automatic branching is used in this course primarily to provide short remedial sequences of additional instruction for students whose performance indicates an inadequate grasp of the principles being presented. These remedial sequences are imbedded at various places in the lessons where appropriate and are frequently used to provide additional practice in algebraic skills that have been inadequately learned. Because of this automatic remediation, different students may receive different exercises in a given lesson.

A student who makes an incorrect response to a single exercise may not need an entire sequence of remedial exercises. He may profit from a single specific corrective message, pointing out the error in his response and allowing him another try at the same problem. This kind of specific correction is used for most exercises in the course. Messages are provided, not for all possible incorrect responses, but for those incorrect responses most likely to occur and most indicative of a student's misconception.

Besides the immediate remediation provided in the specific correction messages and in the imbedded sequences of remedial exercises, opportunity is also provided for review and practice of previously covered topics. This review is given in the form of lesson reviews and block reviews, as described in the following section.

Organization of the Course

The AID course contains a main strand of 50 lessons organized into "lesson blocks." Each lesson block contains five teaching lessons, a test, and a block review. The fiftieth, and last, lesson is a set of programming exercises with no new instruction and is not included in a lesson block. The structure of the lesson strand is illustrated in Figure 1. Lessons numbered 6, 13, 20, 27, 34, 41, and 48 are tests; and lessons numbered 7, 14, 21, 28, 35, 42, and 49 are block reviews. The teaching lessons present concepts in a tutorial style and supply numerous practice exercises; the exercises in the teaching lessons provide hints and allow the student an unlimited number of trials per exercise. The tests are intended to be optional self-tests. They provide no hints and allow the student only one chance at each exercise. The block reviews are also optional and recommended only for those students who do poorly in the preceding tests. These reviews are essentially no more than quick reference sources, providing brief summaries of the topics covered in the lesson block and supplying the student with lesson and exercise numbers for the topics he wants to review. The block reviews for the first two blocks are somewhat more complicated than later block reviews in that appropriate review exercises are automatically selected for the student who wants them, whereas in later blocks students are simply given the reference and expected to use the available student controls to access those exercises.

Each teaching lesson covers a set of related concepts and the length of the lessons varies depending upon the complexity of the ideas covered.

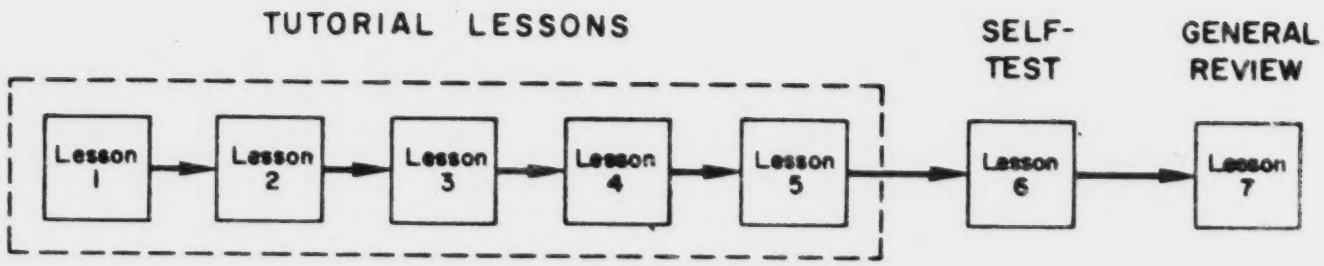


Figure 1. Structure of main lesson strand.

The lessons average about 30 exercises in length with a range from 14 to 63 exercises. The lessons take about an hour apiece although there is considerable variance because of the differences in length and difficulty. Students need not complete a lesson each day; they may work on the same lesson for several days, or they may take several lessons in one day. Ordinarily, the students take the lessons in numeric order but they are not required to do so; if they wish to skip around in the course, they may.

In the first two lesson blocks each teaching lesson is followed by an optional lesson review. These lessons are for students who are having difficulty getting started in the course and cover the same concepts that were presented in the lessons. The instruction in the lesson reviews is given more slowly and carefully, and additional practice problems are provided. A student need not review an entire lesson since the lesson reviews are structured to allow the student to review only those topics he chooses. At the beginning of each review the topics covered in the lesson are listed with essential topics starred and the student chooses which he wants to review and in which order. Since students are allowed to skip about in the course as they please, a student may skip a lesson review and return to it at a later time.

After a student finishes a lesson, he is asked if he wants a summary. Each summary is printed on an 8-1/2" by 11" form that can be torn off and saved by the student as a permanent record. For many lessons there are also a few optional "extra-credit" problems. These problems are substantially more difficult than the exercises given in the lessons and are intended for more capable students. Wherever the AID course is used

as a graded course, we recommend that correct solutions to these problems entitle the student to extra credit. Not every lesson has such extra-credit problems since they are not always appropriate to the subject matter.

The relationship and sequence of lessons, lesson reviews, summaries, and extra-credit problems are illustrated in Figure 2. The number of exercises in each lesson and review, and the number of extra-credit problems for each lesson are shown in Table 1.

The teaching lessons comprise the first five lessons in each lesson block and form the substantive part of the course. As can be seen in Table 1, there are a total of 1943 exercises in the course; of these 1180 are in the teaching lessons. All lessons except the teaching lessons are optional and may be bypassed by a student who wishes to complete the course as quickly as possible. A student who could work at the rate of one problem every two minutes, a reasonable expectation for a good student, could complete the course in 40 hours. Several of the teaching lessons could also be omitted, namely, those on recursive functions, trigonometric functions, and exponential functions. If a student skipped these lessons, he might well finish in 30 hours or less. A slower student, on the other hand, might take all of the optional lessons and problems, and put over 100 hours into the course. In colleges where the AID course is given for credit, it is the usual practice to assign a fixed number of hours of computer time to students regardless of their ability. An assignment of five hours per week for 10 weeks is enough to allow the better students to complete the course while slower students finish only the first 20 or so lessons. The course content is organized

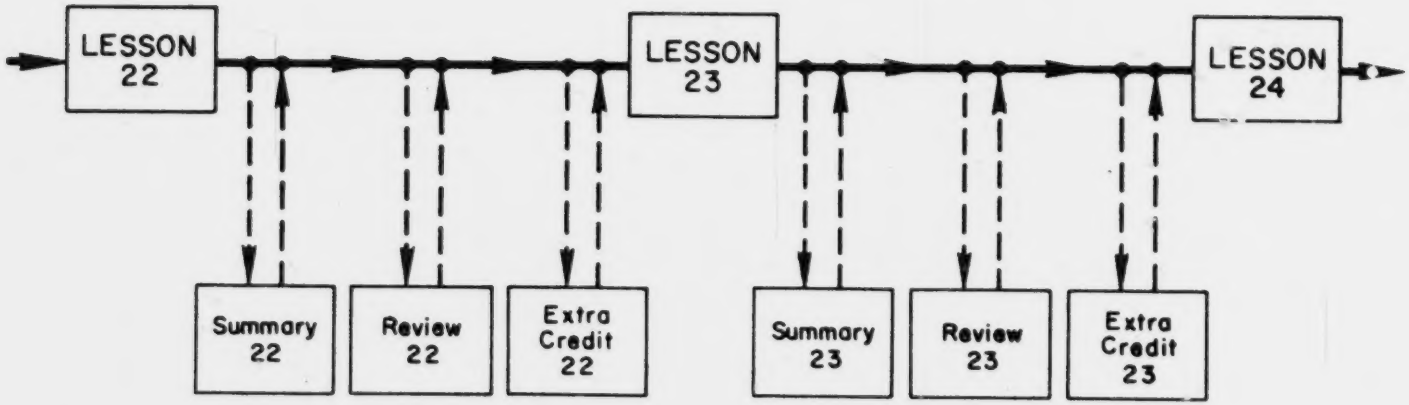


Figure 2. Relationship and sequence of tutorial lessons, summaries, reviews and extra-credit problems.

Table 1

Number of Exercises in Lessons, Lesson Reviews, Tests,
Block Reviews, and Number of Extra-credit Problems

Lesson block	Lesson number	Number of exercises in lesson	Number of exercises in review	Number of extra-credit problems
1	1	18	18	0
	2	28	33	0
	3	49	44	0
	4	61	87	0
	5	63	37	0
	6	43	*	*
	7	13	*	*
Subtotals		219	219	0
2	8	62	23	4
	9	31	28	9
	10	33	40	0
	11	30	25	3
	12	14	14	2
	13	33	*	*
	14	12	*	*
Subtotals		170	130	18

Table 1 (cont'd)

Lesson block	Lesson number	Number of exercises in lesson	Number of exercises in review	Number of extra-credit problems
3	15	62	55	2
	16	27	0	0
	17	27	0	0
	18	35	0	0
	19	32	0	0
	20	27	*	*
	21	9	*	*
Subtotals		183	55	2
4	22	29	0	1
	23	36	0	1
	24	35	0	4
	25	20	0	1
	26	19	0	3
	27	30	*	*
	28	19	*	*
Subtotals		139	0	10
5	29	33	0	2
	30	17	0	2
	31	24	0	0
	32	50	0	2
	33	23	0	2
	34	27	*	*
	35	13	*	*
Subtotals		147	0	8

Table 1 (cont'd)

Lesson block	Lesson number	Number of exercises in lesson	Number of exercises in review	Number of extra-credit problems
6	36	58	0	3
	37	43	0	0
	38	34	0	3
	39	37	0	2
	40	31	0	0
	41	31	*	*
	42	8	*	*
Subtotals		203	0	8
7	43	24	0	0
	44	28	0	0
	45	23	0	2
	46	23	0	1
	47	21	0	1
	48	32	*	*
	49	12	*	*
Subtotals		119	0	4
Totals		1489	404	50

Number of exercises in teaching lessons (first five lessons in each lesson block): 1180

Number of exercises in tests (sixth lesson in each lesson block): 223

Number of exercises in block reviews (seventh lesson in each lesson block): 86

Total number of exercises: 1943

*There are no review or extra-credit problems for tests or block reviews (the last two lessons in each lesson block).

so that the most essential programming concepts are presented in the first half of the course, leaving more advanced concepts such as the conditional definition of functions and the use of matrices to the later lessons. In this way the slower students are given a good introduction to the basic principles of computer science. A brief outline of the lessons is given in Table 2; notice that the first 25 lessons cover algebraic expressions, some standard functions, stored programs, conditional clauses, branching, and loops.

Types of Exercises Used in the AID Course

Although the aim of the AID course is to teach students how to program, and in particular how to program using the AID language, not all of the exercises in the course are programming problems. Most exercises are simpler, ranging in difficulty from trivial true-false exercises to exercises that require the student to construct a complete syntactically and semantically correct AID command that could be used as part of an AID program. The exercises used in the AID course fall into five major groups:

Multiple-choice, and similar, exercises

Short constructed-response exercises

Programming problems

Questions that ask the student to express an opinion or preference

Ungraded exercises

Each of these problem classes is described below, and the first two classes are further subdivided. The exact number of exercises in each of these classes is shown in Table 3 for each teaching lesson and test.

Table 2

Outline of Course

Lesson 1	Using the Instructional Program
Lesson 2	Using AID for Arithmetic
Lesson 3	Order of Arithmetic Operations
Lesson 4	Exponents and Scientific Notation
Lesson 5	The SET and DELETE Commands
Lesson 6	Test of Lessons 1 to 5
Lesson 7	Block Review
Lesson 8	The LET Command
Lesson 9	Some Standard AID Functions
Lesson 10	Indirect Steps, the DO Command, the FOR Clause
Lesson 11	Parts
Lesson 12	The DEMAND Command
Lesson 13	Test of Lessons 8 to 12
Lesson 14	Block Review
Lesson 15	Relations and the Use of the IF Clause
Lesson 16	The TO Command
Lesson 17	Debugging Techniques
Lesson 18	The Indirect Use of the DO Command
Lesson 19	Debugging, Permanent Storage
Lesson 20	Test of Lessons 15 to 19
Lesson 21	Block Review
Lesson 22	The FORM Statement
Lesson 23	Loops
Lesson 24	Loops with Variables in the Exit Condition
Lesson 25	Loops and the FOR Clause
Lesson 26	Loops with a DEMAND Command
Lesson 27	Test of Lessons 22 to 26
Lesson 28	Block Review
Lesson 29	Absolute Value

Table 2 (cont'd)

Lesson 30	SIN and COS
Lesson 31	Exponential and Logarithmic Functions
Lesson 32	Lists
Lesson 33	Using Loops with Lists of Numbers
Lesson 34	Test of Lessons 29 to 33
Lesson 35	Block Review
Lesson 36	Nested Loops and Decrementing Counters
Lesson 37	SUM, PROD, MAX and MIN
Lesson 38	Arrays
Lesson 39	More about Arrays and Lists
Lesson 40	Conditional Definition of Functions
Lesson 41	Test of Lessons 36 to 40
Lesson 42	Block Review
Lesson 43	Recursive Functions
Lesson 44	AND, OR and NOT; Truth Tables
Lesson 45	TV(X) and the FIRST Function
Lesson 46	LET and Boolean Expressions; Debugging Tools
Lesson 47	More Standard AID Functions
Lesson 48	Test of Lessons 43 to 47
Lesson 49	Block Review
Lesson 50	Programming Problems

Class 1: Multiple-choice, and similar, exercises. These exercises pose a set of alternate possibilities that are listed or clearly implied by the text of the exercise. There are four distinct subclasses within this class.

(a) Multiple-choice exercises. This subclass includes only those exercises that are in the traditional multiple-choice format, with the choices explicitly listed and each choice labeled with a letter identifier. A number of multiple-choice exercises in the course have more than one correct choice, and the student is expected to find all of the correct choices. Multiple-choice exercises can be further subdivided according to how many correct choices there are. Also, in some exercises none of the choices listed is correct, and the list of choices includes the entry

N. NONE OF THE ABOVE.

Exercises for which there is no correct true choice could also be put into a separate subclass. The following, finer, division of the class of multiple-choice exercises will not be found in Table 3, which shows only the broader classification.

(i) Multiple-choice exercises with one correct answer (other than exercises in which the correct answer is "None of the above."). Of the 174 multiple-choice exercises in the teaching lessons and tests, 107 are in this class.

(ii) Multiple-choice exercises with two correct choices. There are 39 exercises of this kind.

Table 3

Number of Problems in Each Class of Problems, by Lesson

	Problem Class											
	1a	1b	1c	1d	2a	2b	2c	2d	3	4	5	
Lesson number	Multiple choice	Yes-no	True-false	Other implied choice	Predicted result of AID command	AID command	Reported result of AID command	Other constructed responses	Programming problems	Opinion questions	Ungraded exercises	Total
1	5	2						3		5	3	18
2	11			1	4	2	3		3	4		28
3	11				19	7	3	1	3	5		49
4	7			1	25	1	3	15	3	6		61
5	6				5	9	13	17	9	4		63
6	6		7		12	5	2	6	2	3		43
8		1			32	8	4	8	4	5		62
9	1				17		6		3	4		31
10	4				1	2	4	13	5	4		33
11	1	1		1		6	2	5	6	8		30
12						1	3	2	5	3		14
13	3				13	8		4	2	3		33
15	5		24	6	6		1	6	6	8		62
16	5	1				1		10	2	8		27
17	1							18		4	4	27
18	6							22	1	6		35
19	1	7		3		7		2	4	8		32
20	6	1	1		1	6		10		2		27
22	12	1				1	1	6	4	4		29
23	1					1		28	2	4		36
24	4			1				18	1	7	4	35
25	1							13	1	4	1	20
26	1				1	1		6	2	3	3	19
27	7					2		20		1		30

Table 3 (cont'd)

Problem Class

Lesson number	1a Multiple choice	1b Yes-no	1c True-false	1d Other implied choice	2a Predicted result of AID command	2b AID command	2c Reported result of AID command	2d Other constructed responses	3 Programming problems	4 Opinion questions	5 Ungraded exercises	Total
29					6		3	15	2	5	2	33
30					7	1	2	1	2	4		17
31	5			6	4		3	1	3	2		24
32	4				9	1	2	25	3	3	3	50
33					2	1	2	6	4	5	3	23
34	4		1		8	2		11		1		27
36	3					2	6	31	5	7	4	58
37	1				14	3	4	15	4	2		43
38					11		1	14	2	3	3	34
39	5					1	3	15	6	4	3	37
40	10							13	2	2	4	31
41	11			1	2	6		10		1		31
43	2						9	5	4	3	1	24
44	4		8		1			5	3	2	5	28
45	2				2		3	6	4	3	3	23
46	4				3	1	3	2	4	3	3	23
47	1							15	1	3	1	21
48	13		7					10		2		32
Total, Teaching Lessons:	124	13	32	19	169	57	86	362	113	155	50	1180
Total, Tests:	50	1	16	1	36	29	2	71	4	13	0	223
Totals:	174	14	48	20	205	86	88	433	117	168	50	1403

(iii) Multiple-choice exercises with three or more correct choices. There are 21 exercises in this class.

(iv) Multiple-choice exercises to which the correct response is "None of the above." There are seven exercises like this (although there are many exercises that list "None of the above" as a possibility).

(b) Yes-no questions. The exercises in this class are those in which the two possibilities, yes and no, are implied by the grammatical form of the question rather than being explicitly listed as are the choices for multiple-choice exercises. All of the exercises in this class deal with matters of fact, rather than opinion. A large number of yes-no questions are also found in the class of exercises that ask for students' opinions (Class 4, described below), but those exercises are also not included here.

(c) True-false exercises.

(d) Other "implied-choice" exercises. There are a number of exercises in the course that appear to be constructed-response exercises in that the text does not list the possible answers from which to choose. These exercises are revealed by closer inspection to be more related to multiple-choice exercises than to constructed-response exercises since a limited number of choices are clearly implied in the statement of the problem. As an example, the following question gives two alternatives, one of which is the correct answer:

IF YOU USED THIS COMMAND

TYPE 1/100

WOULD AID GIVE THE ANSWER IN DECIMAL FORM OR IN

SCIENTIFIC NOTATION?

Class 2: Short-constructed-response exercises. These exercises require short-constructed responses that will be checked for correctness by the instructional program. Class 3 exercises and Class 5 exercises, described below, could also be considered constructed-response exercises and are distinguished from exercises in this class in that they are not checked for correctness by the instructional program, for reasons given below. The short-constructed-response exercises, like the multiple-choice exercises, can be further divided into several subclasses, which are described below and are tallied in Table 3.

(a) Exercises that ask the student to predict the result of using a given set of AID commands. In these exercises the students are shown an AID command or sequence of commands and are asked to predict what result would be given if such commands were used.

(b) Exercises that require the student to construct an AID command. In these exercises the student's response is analyzed by the instructional program, as are all exercises in Class 2; students are also expected to construct AID commands as they work the programming problems in Class 3, but those commands are not analyzed by the instructional program, and are not included in this class. On occasion students are asked to construct a

part of an AID command or to complete a given partial command; those exercises are in Class 2d, below.

(c) Exercises that ask the student to report results obtained from student-constructed AID programs. These exercises always follow an exercise from Class 3. They are used by the instructional program to judge the correctness of the program written by the students for the preceding Class 3 exercise. A more complete explanation of the sequence of problems will be given below in the description of the Class 3 exercises.

(d) Other constructed-response exercises. This class incorporates the miscellaneous constructed-response exercises. Exercises similar to those in Class 2b but requiring the student to construct only a part of an AID command, rather than an entire command, are included here. Also included are exercises similar to those in Class 2c but requiring non-numeric responses. The class is large and heterogenous; the only common characteristics of the exercises in this class are that they all require constructed responses less than one line long that are checked for correctness by the instructional program.

Class 3: Programming problems. Characterization of the exercises in this class as "programming problems" is imprecise; many of these exercises require the student to construct only a single AID command that is not part of a program and may simply be copied from the text given by the instructional program. However, the class is well-defined in that all its exercises require the student to use the computer as a programmer,

not as a student. To clarify this issue a few remarks about the instructional system are called for.

The various types of commands that can be written in the AID language and used in AID programs are introduced to the student one at a time, using a sequence of instruction like the one described below.

First, an example of an AID command or program is shown and its use explained.

Second, the student is shown several examples and asked to predict the result of using such a command or program.

Third, the student is required to construct part or all of a similar command or program.

Fourth, one or more programming problems illustrating the use of the new principle are given.

In the first three steps of the sequence described above the instruction is given to the students by means of an instructional program named INST; in the fourth step the students use the AID interpreter which interprets and executes AID commands. After using the AID interpreter to solve the problem, students return to the instructional program for further instruction. Programming problems are posed by the instructional program, but once a student starts using the AID interpreter he is given no further instruction until he branches back to INST. This branching is completely under student control; the details of accessing both programs are taught in the course. Thus, while the student is using AID he is completely on his own, communicating with the computer by means of the AID interpreter, just as a working programmer would do. These two programs, INST and AID, together control all of the student's

interaction with the computer as he takes the AID course, and together form the interactive part of the instructional system. In essence, the difference between these two programs is that INST talks about the language AID, while the AID interpreter uses the language AID. The AID interpreter is a commercial program that was written for the use of programmers, not students, and contains no routines for comparing a student's program to a correct solution for the same problem. For instructional purposes, however, it is desirable to know if a student solved the given problem, and this is accomplished in the course by the instructional program which keeps track of which programming problem the student is working on and asks him about the results obtained by his program after he returns to the instructional program. The problems that the student is working on while he is using the AID interpreter are exactly the problems in Class 3, and the exercises that ask the students the results of his work are largely in Class 2c, although a few are in Class 2d.

As mentioned before, not all of the exercises in Class 3 should be labeled programming problems since some require no more than a single direct command. Thus, the difficulty of exercises in this class varies from the easiest to hardest to be found in the course. Some exercises, in fact, are no more than trivial copying tasks (with the added complication of starting and stopping the AID interpreter), while others are programming problems of a complexity that might challenge experienced programmers. The exercises in Class 3 could be further subdivided into at least three subclasses on the basis of difficulty. The easiest subclass, which asks the student to start the AID interpreter, copy a given

set of commands, and observe the result, contains 37 of the 117 programming problems in the teaching lessons. Another seven programming problems are almost as simple, requiring only minor modifications of programs given as examples in the lessons. There are 73 problems that are difficult enough to require some original thinking or problem-solving skills on the part of the student; these 73 problems themselves vary considerably in difficulty. Note that we are talking here about the 117 programming problems in the teaching lessons and tests. There are also numerous programming problems in the review lessons, and all of the extra-credit problems are programming problems.

Class 4: Opinion questions. These exercises ask the students to express an opinion or preference. Typical exercises of this class are "Would you like to review any of the topics from Lesson 8?", "Do you want the summary for this lesson?", and "Did your program give the results you expected?" The response to these exercises are analyzed, and acted upon, by the instructional program, but are not classified as right or wrong. Most of these exercises are in the form of yes-no questions.

Class 5: Ungraded exercises. This class of exercises is very small and contains only those exercises that require responses that cannot for one reason or another be graded by the instructional program. These exercises are separate from the programming problems in Class 3 which cannot be graded for an entirely different reason. Some of the ungraded exercises are ungraded, not because of any theoretical difficulty, but simply because the requisite mechanism does not exist in the instructional program at this time. Others are ungraded because they ask for freely constructed responses that cannot be analyzed because of the complexity

of the English language; in these cases, students are given a sample correct answer and asked to judge for themselves whether or not they responded correctly (no record of the student response is kept by the system).

The above scheme for classifying the exercises in the course is necessarily a hybrid scheme, using both response format and exercise content as bases for classification. This scheme was chosen because it gives a good picture of the curriculum. Other schemes could also have been used. One possible scheme of classification would be to divide the exercises strictly according to the form of the expected response. This classification would yield classes such as

multiple-choice exercises

yes-no questions

true-false exercises

constructed-response exercises: numeric

constructed-response exercises: single letter or character

constructed-response exercises: word or phrase

constructed-response exercises: AID command

constructed-response exercises: AID program

This classification has certain virtues, one of which is that there is no ambiguity, but distinctions such as the one between "opinion" questions and other yes-no questions are lost.

Another method of classification that would yield a different profile of the course would depend strictly upon content and not upon the form of the response. Since the division of the course into lessons is a division based upon content, a closer look at the content of each lesson is warranted.

Description of Content of Lessons

A complete table of contents for the teaching lessons is given in the appendix which lists all the topics discussed in each lesson together with a list of the exercises on that topic (and the number of exercises for each topic). Below is a more informal discussion of the teaching lessons, tests, and block reviews, with numerous examples.

Lesson 1: Using the instructional program. Lesson 1 is a set of 18 exercises explaining how to use the instructional program. The mechanics of typing and erasing responses are explained, and instructions are given for starting and stopping the program. The student is also taught how to get additional instruction (hints), how to get the answer for any exercise from the program, and how to control the sequence of instruction.

The style of instruction in Lesson 1, as in succeeding lessons, is informal.

Lesson 1, Exercise 3:

IF MULTIPLE CHOICE PROBLEMS HAVE MORE THAN ONE CORRECT ANSWER YOU CAN LIST THE CORRECT CHOICES IN ANY ORDER. SUPPOSE B, C, AND D ARE THE CORRECT CHOICES FOR A PROBLEM. WHICH OF THESE WOULD BE CORRECT WAYS TO ANSWER?

A. D, B, C, A

B. B, D, C

C. B, C, D

D. D, B, C

Lesson 1, Exercise 14:

FROM LESSON 1, YOU SHOULD HAVE LEARNED HOW TO SIGN ON AND OFF, HOW TO START AND STOP THE TEACHING PROGRAM, HOW TO GET A HINT, AND HOW TO USE CTRL-G. DO YOU WANT TO REVIEW ANY OF THESE TOPICS?

In Lesson 1, five of the 18 exercises are multiple choice, similar in form to Exercise 3 shown above. This proportion of multiple-choice exercises is fairly typical of the course.

Exercise 14 illustrates an instructional strategy that is used in the lessons in the first two lesson blocks. At the end of each lesson, its content is briefly summarized and the student is asked if he wants to take the lesson review. In this way the student is forced to judge whether he is competent to proceed with the course or whether he needs additional instruction and practice.

A number of exercises are designed more to elicit opinion than information, and have no "correct" answer. Exercise 14 above exemplifies these; other examples are questions like "Do you want to go on to Lesson 2 now?" and "Do you want to practice signing on and off?" In Lesson 1 there are five exercises of this type (Class 4).

The exact number of each type of exercise in Lesson 1 (and other teaching lessons and tests) is shown in Table 3.

Lesson 2: Using AID for arithmetic. In Lesson 2 the student gets his first experience with the AID interpreter. The 28 exercises in this lesson teach the student how to start and stop the AID interpreter and how to use the AID interpreter for doing simple arithmetic by giving direct "TYPE" commands. The AID symbols for the four simple arithmetic

operations (+, -, *, and/) are taught, and the use of optional parentheses in arithmetic expressions is introduced. By the end of the lesson the student is able to start the AID interpreter and give simple, direct commands like

TYPE 5/25

TYPE 3.25 + 17.4 + 3.12

TYPE 15*17 + 25*19

One of the most persistent errors made by students learning an algebraic programming language is the omission of the multiplication operator in certain kinds of algebraic expressions. The root of this difficulty is the convention used in ordinary algebra of implying multiplication by juxtaposition. For example, we ordinarily write

$24x + 56(y - z)$

without explicit multiplication operators. AID, like other algebraic programming languages, requires the use of operation symbols:

24*X + 56*(Y - Z)

In Lessons 2 through 5 there are a number of exercises aimed specifically at preventing the error of omitted operation symbols.

Lesson 2, Exercise 11:

WHICH ARE VALID AID COMMANDS?

A. TYPE (17.01)/32.765

B. TYPE 1/2 + .1785 - (12/16)

C. TYPE 2(10) + 3(10) + 4(10)

D. TYPE 1/2 ÷ (7* 3/2)

N. NONE OF THE ABOVE

Lesson 2, Exercise 19:

USE AID TO DO THESE PROBLEMS:

1. FIND THE AREA OF A RECTANGLE WITH WIDTH 1.72375 AND LENGTH 12.001325.
2. SUPPOSE A SQUARE OF WIDTH .63725 IS CUT FROM THE ABOVE RECTANGLE. FIND THE AREA OF THE SQUARE.
3. FIND THE AREA OF THE REMAINING PART OF THE RECTANGLE.

Of the 28 exercises in Lesson 2, 12 are constructed-response exercises. These exercises vary in difficulty with the most difficult being Exercise 19, shown above. None of these exercises approaches in difficulty the programming problems given later in the course.

Lesson 3: Order of arithmetic operations. The arithmetic used in Lesson 2 was relatively simple, but in Lesson 3 the complexity increases with the addition of the concept of hierarchy of operations and the use of parentheses. Because of the necessarily linear nature of computer input, an algebraic formula cannot be written on more than one line. The following expression, for example, is typically written on two lines.

$$\frac{x + y}{x - y}$$

In AID this expression would be transformed into

(X + Y)/(X - Y)

The horizontal bar is replaced with a slash and one of the functions of the bar, that of implied grouping, is lost, so that parentheses must be added. The latter expression is not only more difficult to read, but also more difficult to construct correctly since it requires the student to make a conscious decision about the desired order of evaluation. A

large part of Lesson 3 is devoted to teaching the student how to force the order of evaluation by the use of parentheses, and how to determine the order of evaluation if parentheses are not used, by using an explicit set of rules for the hierarchy of the four basic arithmetic operations.

Lesson 3, Exercise 10:

LOOK AT THESE THREE COMMANDS. AID WILL GIVE THE SAME ANSWER TO TWO OF THEM. WHICH TWO?

TYPE 3 + (2*4)

TYPE (3+2) * 4

TYPE 3 + 2 * 4

START AID AND TRY THE THREE COMMANDS.

Lesson 3, Exercise 22:

TYPE 100/10/10/2

COULD BE WRITTEN AS

A. TYPE (100/10)/(10/2)

B. TYPE (100/(10/10))/2

C. TYPE (100/(10/10/2))

D. NONE OF THE ABOVE

Since a large part of Lesson 3 reviews algebraic notions that may be better understood by some students than by others, several remedial sequences are imbedded in the lessons. Since these remedial sequences are bypassed by students who are responding correctly, a good student can complete Lesson 3 in 27 exercises whereas a student who performs poorly may receive up to 49 exercises.

Lesson 4: Exponents and scientific notation. Lesson 4 is much longer than average (61 exercises), and extends the work on arithmetic

expressions to include expressions with exponentiation. First, the concept of exponentiation is reviewed, and the AID symbol (↑) is introduced. The rules for the hierarchy of operations are extended to include exponentiation, and the AID form of scientific notation is introduced. Negative exponents, fractional exponents, and the zero exponent are also covered. Lesson 4, like Lesson 3, is largely review of algebraic principles that may have been forgotten. The exercises also provide practice in reading and constructing expressions in the linear form required by the AID interpreter.

As noted above the horizontal division bar so frequently used in algebra has two functions, that of signifying division and that of implying grouping. The usual notation for exponentiation also has two functions: signifying exponentiation and implying grouping. As an example, the following expression raises number 5 to the power 1/2

$$5^{1/2}$$

However, when this expression is translated into an AID expression with the symbol ↑ used to denote exponentiation, the grouping function is lost and parentheses must be added:

```
5↑(1/2)
```

Since students are accustomed to this grouping function of ordinary notation, they may erroneously translate the expression above into

```
5↑1/2
```

which is equivalent to

$$5^{1}/2$$

Several exercises in Lesson 4 are designed to prevent errors of this kind.

Lesson 4, Exercise 2:

WHAT WOULD AID ANSWER TO THIS COMMAND?

TYPE 2↑3

Lesson 4, Exercise 12:

USE AID TO EVALUATE EACH OF THE FOLLOWING.

1. 4 SQUARED TIMES 3.1416
2. THE SUM OF 4 CUBED AND 6
3. THE SUM OF THE SQUARES OF 1, 2, 3, 4, 5, 6, 7, AND 8

In Lesson 4, 47 of the 61 exercises require constructed responses. Most of these are numeric results of arithmetic calculations to answer questions like that in Exercise 2. Three exercises, with several parts, require the student to use the AID interpreter, and he is encouraged to use AID throughout the lesson whenever he wishes.

Lesson 5: The SET and DELETE commands. After the sizeable dose of arithmetic given in Lessons 3 and 4, Lesson 5 returns to the mainstream of instruction with the introduction of two new AID commands: the SET command and the DELETE command. SET is used in AID to assign numeric values to variables, and DELETE is used to delete a previous assignment or definition. In AID, variables are single letters, and the forms of the SET and DELETE commands are straightforward and easily learned (SET X = 5 + 2, DELETE X). A few word problems are given to illustrate the use of the new commands.

Lesson 5 also introduces the multiple-argument form of the TYPE command, in which several TYPE commands can be combined by separating the arguments with commas (TYPE X, Y, X + Y).

Lesson 5, Exercise 3:

WHAT WILL AID ANSWER AFTER THESE COMMANDS?

SET B = 1.5

TYPE 3*B

Lesson 5, Exercise 11:

WHAT COMMAND WILL CAUSE AID TO SET M EQUAL TO S PLUS 9?

Lesson 6: Test of Lessons 1 to 5. Lesson 5 is the last teaching lesson in the first lesson block, and is followed by a test in Lesson 6. Like other tests, Lesson 6 supplies no hints, and students are allowed only one try on each exercise. However, correct answers are programmed for each exercise, and the student may request these at any time. Whenever a student misses an exercise, he is given a review reference and advised to review that topic before proceeding with the course.

The exercises in Lesson 6 are classified by type in Table 3. Of the 43 exercises in Lesson 6, three ask for students' opinions and two are programming problems that require the use of AID and are not directly checked by the instructional program. The remaining 38 exercises are test exercises that are checked by the instructional program; these 38 exercises are classified in Table 4 according to which of the teaching lessons in the lesson block are being tested. Many of the exercises test more than one lesson since the material being taught builds progressively from one lesson to the next; such exercises are listed only once in Table 4 however.

Lesson 7: Block review (general review of Lessons 1 to 5). At the beginning of Lesson 7 the student is informed that the block review is optional, but recommended for students who missed more than five problems

Table 4

List of Exercises that Test Each Teaching Lesson

Number of teaching lesson	Test Exercises	
	Test number	Exercise numbers
1	6	2, 3, 4, 5, 6, 7, 9, 10, 12, 13, 14, 15, 16
2	6	8, 11, 17, 18, 19, 20, 38.1, 39.1
3	6	21, 22, 23
4	6	24, 25, 26, 27
5	6	28, 29, 30, 31, 32, 33, 34, 35, 36, 37
8	13	1.1, 2, 3, 4, 5, 6, 7, 8
9	13	9, 10, 11, 12, 13, 14, 16
10	13	17, 18, 19, 20, 21, 22, 23, 24
11	13	25, 26, 27
12	13	28, 29.1
15	20	1.1, 2, 3, 4, 5, 6
16	20	7, 16
17	20	8, 9, 10
18	20	11, 12, 13, 14, 15
19	20	17, 18, 19, 20, 21, 22, 23, 24, 25
22	27	1, 2, 2.1, 2.2, 2.3, 3, 4
23	27	5, 5.1, 5.2, 5.3, 5.4, 10
24	27	6, 6.1, 6.2, 6.3, 7, 7.1, 8
25	27	5.5, 9, 11, 11.1, 12, 13, 14
26	27	15, 15.1
27	34	1, 2, 3, 4, 5, 6, 7
30	34	8, 9, 10, 11

Table 4 (cont'd)

Number of teaching lesson	Test Exercises	
	Test number	Exercise numbers
31	34	12, 13, 14, 15, 15.1, 16, 17
32	34	18, 18.1, 18.2, 19, 20, 21, 22, 23
33	34	None
36	41	1, 2, 3, 4, 5, 6
37	41	7, 8, 9, 10
38	41	11, 12, 13, 14, 15, 16
39	41	17, 18, 19, 20, 21, 22, 23
40	41	24, 25, 26, 27, 28, 29, 30
43	48	1, 2, 3, 4, 5
44	48	6, 7, 8, 9, 10, 11, 12
45	48	13, 14, 15, 16, 17, 18
46	48	19, 20, 21, 22, 23, 24, 25, 26
47	48	27, 28, 29, 30

in the preceding test (Lesson 6). In order to allow students to review only selected portions of the lessons in the lesson block, the branching structure used in Lesson 7 is more complex than that used in the teaching lessons and tests. The individual lesson reviews for Lessons 1 to 5 are called as subroutines by Lesson 7, and students may select not only the lesson they want to review but a particular part of the lesson. The following example from Lesson 7, Exercise 5 illustrates how this selection procedure operates.

> LESSON 4 WAS ABOUT EXPONENTS AND SCIENTIFIC NOTATION.
> FRACTIONAL EXPONENTS AND NEGATIVE EXPONENTS WERE DISCUSSED,
> AND ALSO THE USE OF 0 AND 1 AS EXPONENTS. THE ORDER OF
> ARITHMETIC OPERATIONS + - * / AND ↑ WAS COVERED.
> DO YOU WANT TO REVIEW ANY OF THESE THINGS?

If a student answers "yes" to the above question, he is sent to the review lesson for Lesson 4, where he is allowed to review any of the topics in Lesson 4 in any order he wants.

This first block review also reminds students that they can control the sequence of instruction by using the CTRL-G key and that they can use the student manual as a reference book for the course.

Lesson 8: The LET command. Lesson 8, the first lesson of the second lesson block, introduces the LET command, used in AID to define functions. The syntax of the LET command for functions of one, two, and three variables is taught, as is the syntax of function calls. The difference between LET and SET commands is explained and the use of the DELETE command for deleting function definitions is described. A substantial

part of Lesson 8 is on substitution of arithmetic expressions for variables in function calls and other arithmetic expressions.

Lesson 8, Exercise 14:

WHAT WILL AID ANSWER?

LET Q(A, B, C) = C*(A + B)/2

TYPE Q(3, 5, 7)

Lesson 8, Exercise 28:

USE AID TO DO THIS PROBLEM. DEFINE A FUNCTION TO CONVERT DEGREES FAHRENHEIT TO DEGREES CENTIGRADE. THEN CONVERT THESE TEMPERATURES TO CENTIGRADE:

0, 10, 32, 72, 212

Lesson 9: Some standard AID functions. In Lesson 9, the student is introduced to four of the standard AID functions. These are:

SQRT(X) - the square root function;

IP(X) - the "integer part" function;

FP(X) - the "fraction part" function;

SGN(X) - the sign function.

These functions, together with functions defined by the student, are used in several programming problems.

Lesson 9, Exercise 14:

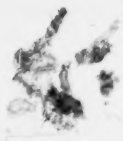
YOU CAN USE THE AID FUNCTION FP(X) TO FIND OUT IF ONE NUMBER CAN BE DIVIDED BY ANOTHER WITHOUT A REMAINDER...

IS 2976 EVENLY DIVISIBLE BY 3?

Lesson 9, Exercise 16:

THE SIGN FUNCTION

SGN(X)



GIVES 1 IF X IS A POSITIVE NUMBER

AND 0 IF X IS 0

AND -1 IF X IS A NEGATIVE NUMBER

WHAT WILL AID ANSWER?

TYPE SGN(25)

Lesson 10: Indirect steps, the DO command, the FOR clause. In Lesson 10 the concept of a stored program is introduced. Up to this point, students have been using AID as a desk calculator, doing all exercises with direct commands, i.e., commands that are executed immediately. In this lesson students are taught that TYPE commands can be stored for later execution by prefacing the command with a "step number," as in the following examples:

2.1 TYPE F(16)

4.7 TYPE X↑2, X↑3

They are also taught how to execute these stored commands by using a DO command.

Two variants of the FOR clause are used to modify DO commands. In the first variant, values for the iteration variable are given by a simple listing:

DO STEP 17.3 FOR Y = 1, 2, 7, 14.3

In the second variant of the FOR clause the values for the variable are given in a range specification, which gives an initial value for the variable, a step size, and a final value:

DO STEP 5.6 FOR X = 3(2)9

This command specifies that X will assume the value 3, then be incremented

by 2 after each iteration of step 5.6 until X $\geq$ 9. This is equivalent to the FORTRAN form:

```
      DO 56 X = 3, 9, 2
   56 < statement >
```

and the ALGOL form:

```
      FOR X ← 4 STEP 2 UNTIL 9 DO < statement >
```

Lesson 10, Exercise 6:

START AID AND GIVE THESE COMMANDS:

```
47.3 TYPE X, X↑2, X↑3, X↑4, X↑5
SET X = 3
DO STEP 47.3
SET X = 4
DO STEP 47.3
SET X = 5
DO STEP 47.3
```

WHAT IS THE VALUE OF X↑5 IF X = 4?

Lesson 10, Exercise 17:

WHAT VALUES OF A WILL BE USED IF THIS COMMAND IS GIVEN?

```
DO STEP 73.7 FOR A = 5(10)35
```

A. 5, 10, 15, 20, 25, 30, 35

B. 10, 15, 20, 25, 30, 35

C. 5, 15, 25, 35

Lesson 11: Parts. Lesson 11 explains how indirect (stored) steps are grouped into parts. Steps 12.1, 12.7, and 12.8, for example, are grouped as "Part 12," and can be executed by a single command:

```
DO PART 12
```

The sequence of execution depends only upon the numerical order of the step numbers, and not upon the sequence in which they were written. Thus, steps 3.7, 3.2, and 3.8 will be executed in the order 3.2, 3.7, 3.8. Students have little difficulty with this concept except when step numbers end in zeros; some students cannot readily sort into numerical order a sequence like 3.5, 3.8, 3.10 (the correct order is 3.10, 3.5, 3.8).

Lesson 11, Exercise 5:

```
YOU CAN TYPE THE STEPS IN ANY ORDER, BUT AID WILL ALWAYS
DO THEM IN NUMERICAL ORDER.  WHICH STEP WILL BE DONE FIRST?
    17.4  TYPE X* Y
    17.5  SET N = 5
    17.2  SET X = 10
    17.3  SET Y = 2
```

Lesson 11, Exercise 11:

```
A PART (SET OF INDIRECT STEPS) IS ALSO CALLED A PROGRAM.
USE AID TO WRITE A PROGRAM THAT WILL LIST THE RADIUS,
DIAMETER, CIRCUMFERENCE, AND AREA OF A CIRCLE OF RADIUS R.
THEN USE THE PROGRAM FOR R = 10, 20, 30, 40, AND 50.
```

Lesson 12: The DEMAND command. In Lesson 12 the DEMAND command is introduced. The DEMAND command is used in AID programs for keyboard input. The DEMAND command can be used only indirectly, unlike previously introduced commands which can be used both directly and indirectly. DEMAND is used for numerical input only and the form is simple:

```
DEMAND X
```

where X is the variable name to which the input number is assigned.

Lesson 12 also introduces the TIMES clause which can be used to modify a DO command, in this way:

DO PART 7, 5 TIMES

Lesson 12, Exercise 4:

START AID AND WRITE A PROGRAM THAT WILL ASK YOU FOR 3 NUMBERS, A, B, AND C, AND THEN GIVE YOU THE AVERAGE OF THE 3 NUMBERS. AFTER YOU HAVE TESTED YOUR PROGRAM, USE IT TO FIND THE AVERAGE OF

A = 179.053

B = 23.7

C = 271.0015

Lesson 12, Exercise 5:

WHAT COMMAND WOULD YOU USE IF YOU WANTED PART 2 DONE 7 TIMES?

Lesson 13: Test of Lessons 8 to 12. This lesson is the test for the second lesson block and is structured like other tests: the student is given only one try for each exercise, and no hints are provided although a student who cannot do an exercise can request the correct answer.

As for the first test (Lesson 6), the exercises are classified by type in Table 3, and are classified according to which of Lessons 8 through 12 are being tested in Table 4. An exercise that might be considered a test of more than one lesson is listed only once in Table 4.

Of the 33 exercises in Lesson 13, only 28 are properly test exercises, since three ask for opinions (Class 4 exercises) and two ask the

student to use the AID interpreter but do not check the work done by the student while he is using AID.

Lesson 14: Block review. Lesson 14, like other block reviews, is optional; the review is recommended for students who missed more than three problems in the preceding test.

Lesson 14, Exercise 3:

> LESSON 8 WAS ABOUT THE "LET" COMMAND AND HOW TO USE IT TO DEFINE A FUNCTION. FUNCTIONS OF 2 AND 3 VARIABLES WERE DISCUSSED. INSTRUCTIONS FOR PRINTING AND DELETING A FUNCTION WERE GIVEN.
>
> DO YOU WANT TO REVIEW ANY OF LESSON 8?

The student who answers "yes" will be branched to the lesson review for Lesson 8 and then given his choice of which topics to review in what order.

Lesson 15: Relations and the use of the IF clause. Lesson 15 begins a new lesson block with the introduction of the most powerful of programming tools, the conditional clause. The conditional (IF) clause consists of the word "if" followed by a Boolean statement, and may be appended to any of the commands so far introduced.

```
SET Z = 2 IF X < 10
TYPE X IF X > 0
DO PART 5 IF M = N
```

Most of Lesson 15 concerns the syntax and meaning of logical statements. The following AID symbols for arithmetic relations are introduced:

< less than;

> greater than;

<= less than or equal;

>= greater than or equal;

not equal

The terms "positive," "negative," and "non-negative" are reviewed. The Boolean connectives "and" and "or" are also introduced although their meanings and the hierarchy for the connectives are not discussed extensively at this point. The students are required to write several programs using conditional clauses.

Lesson 15, Exercise 14:

STUDY THIS PROGRAM.

49.5 TYPE X IF X > Y

49.6 TYPE Y IF X <= Y

DO PART 49

IF X = 12.1 AND Y = 6, WHAT WILL AID ANSWER?

Lesson 15, Exercise 15:

USING AID, WRITE A PROGRAM THAT WILL FIND THE SMALLER OF TWO NUMBERS X AND Y. TRY SEVERAL DIFFERENT VALUES OF S AND Y.

Lesson 16: The TO command. Lesson 16 introduces the idea of conditional branching and provides additional practice in the use of conditional clauses. Although Lesson 16 is quite short (27 exercises), some of the programming problems are very difficult. Several sample programs are analyzed in detail with special emphasis on the order of execution.

Lesson 16, Exercise 3:

HERE IS A PROGRAM THAT CALCULATES THE AREA OF A RECTANGLE OF LENGTH L AND WIDTH W. IF EITHER L OR W IS NEGATIVE, PART 15 IS USED TO GIVE AN "ERROR" MESSAGE.

```
14.1  DEMAND L
14.2  TO PART 15 IF L < 0
14.3  DEMAND W
14.4  TO PART 15 IF W < 0
14.5  TYPE L*W
15.1  TYPE "DO NOT USE NEGATIVE NUMBERS."
```

WHICH STEPS WILL BE DONE IF L = 5 AND W = -3?

Lesson 16, Exercise 6:

WRITE A PROGRAM THAT WILL TYPE 3 NUMBERS A, B, AND C IN NUMERIC ORDER (THAT IS, THE SMALLEST FIRST, ETC.)

Lesson 17: Debugging techniques. Lesson 17 concentrates on the debugging technique of tracing the step-by-step execution of a program listing the changes in values of the variables used in the program.

Lesson 17, Exercise 3:

FOR PRACTICE, LET'S MAKE A TRACE OF THIS PROGRAM, ASSUMING A = 3.

```
31.3  DEMAND A
31.2  SET B = A↑2 - 10
31.3  SET C = A IF A > B
31.4  SET C = B IF A <= B
31.5  TYPE B
31.6  TYPE C
```

FILL IN THE VALUES OF C IN THIS TRACE (STARTING AT STEP 31.3).

STEP	A	B	C
31.1	3	-	-
31.2	3	-1	-
31.3	3	-1	?
31.4	3	-1	?
31.5	3	-1	?
31.6	3	-1	?

Lesson 18: The indirect use of the DO command. In Lesson 18 the indirect use of the DO command is introduced. Up to this point the student has used DO commands directly to execute programs or single steps. The DO command can also be used indirectly, as part of a stored program, or to execute subroutines. Frequently, a conditional clause is appended to indirect DO commands.

Several exercises in Lesson 18 provide the student with practice in determining the order of execution of steps in a program with conditional subroutine calls.

Lesson 18, Exercise 2:

WHEN AID COMES TO AN INDIRECT "DO" COMMAND, IT WILL DO THE STEP OR PART INDICATED AND THEN RETURN TO THE STEP AFTER THE "DO" COMMAND.

```
16.1  DO STEP 2.1 IF Q < 0
16.2  TYPE Q
 2.1  SET Q = -Q
      DO PART 16
```

IF Q = 3, THE STEPS WILL BE DONE IN WHICH ORDER?

A	B	C	D
16.1	16.1	16.1	16.1
2.1	16.2	2.1	16.2
16.2	2.1		

Lesson 19: Debugging, permanent storage. The first part of Lesson 19 is an optional section of tips for writing and debugging programs. This section is primarily for students who have difficulty with the programming problems in the preceding lessons and covers such topics as planning and editing a program, distinguishing between and correcting several kinds of syntactic and semantic errors, and single-stepping through a program with commands like

DO STEP 34.2.

The second half of the lesson describes the difference between core memory and disk storage, and teaches students how to store their programs on disk by using the AID file commands: USE, FILE, RECALL, and DISCARD.

Lesson 19, Exercise 5:

SUPPOSE YOU FORGOT TO TYPE ONE OF THE STEPS IN A PROGRAM. IF THERE IS A STEP MISSING BETWEEN STEPS 2.5 AND 2.6, YOU CAN INSERT THE STEP BY USING WHAT STEP NUMBER?

Lesson 19, Exercise 13:

WHAT COMMAND WOULD YOU USE TO FILE PART 29 AS ITEM 3?

Lesson 20: Test of Lessons 15 to 19. This lesson is the test for the third lesson block. Lesson 20 contains 27 exercises of which two are requests for opinions. The other 25 are grouped in Table 4 according to which of the preceding lessons they test.

Lesson 21: Block review. Lesson 21 is a review of the lessons tested by Lesson 20. Like other block reviews it is optional, but is recommended for students who miss more than three problems in the test. Unlike preceding block reviews this one does not call on lesson reviews as subroutines, since there were no lesson reviews for Lessons 16 to 19; instead, the students are given references to pertinent exercises in the lessons themselves.

Lesson 21, Exercise 8:

> LESSON 19 EXPLAINED HOW TO PLAN, WRITE, AND EDIT A PROGRAM; WHAT KINDS OF ERRORS THERE ARE AND HOW TO CORRECT THEM; AND HOW TO USE PERMANENT STORAGE.
>
> DO YOU WANT TO REVIEW ANY OF LESSON 19?

A student who answers "yes" will be given a list of the topics in Lesson 19 and the exercises that treat each of those topics.

Lesson 22: The FORM statement. The FORM statement, used in AID to specify the form of teletype output, is introduced in Lesson 22. Up to this point students use the standard AID form for teletype output, but in this case they learn how to define new forms for themselves. A FORM statement allows the programmer to determine the spacing used in the output, to define positions for more than one number per line, and to insert text into the output. The number of digits to be printed can be specified in a FORM statement and any numbers printed will be rounded to fit the space specified. The FORM statement can also be used to type text, although a TYPE command can equally well be used for this purpose, as explained in the lesson.

Lesson 22, Exercise 16:

> WRITE A PROGRAM (PART 7) THAT WILL GET A VALUE OF X FROM THE USER AND THEN TYPE X, X SQUARED, X CUBED, AND X TO THE FOURTH POWER ON ONE LINE, LIKE THIS:
>
> X = 3.0 X↑2 = 9.0 X↑3 = 27.0 X↑4 = 81.0
>
> SAVE THIS PROGRAM BY TYPING "USE FILE 100" AND "FILE PART 7 AS ITEM 1" BECAUSE YOU WILL NEED IT FOR ANOTHER PROBLEM LATER.

Lesson 22, Exercise 19:

> START AID AND CHANGE THE PROGRAM YOU WROTE FOR PROBLEM 16 SO THAT IT PRINTS THE TITLE "POWERS OF X". ALSO PUT 1 OR 2 BLANK LINES AFTER THE TITLE. TRY THE PROGRAM FOR X = 1.2, .7, 6.25.

Lesson 23: Loops. Lesson 23 introduces the vital subject of loops. Several later lessons will also deal with this topic, and the loops used in this lesson are of the simplest sort. These loops use an incremented variable to count the number of times the loop is used. The students are taught to set the counter to an initial value before the loop, to add a constant to the counter inside the loop, and to end the loop with a conditional branch in which the value of the counter is compared to a constant.

The lesson starts with a detailed study of several sample programs incorporating simple loops. Several exercises are intended to illustrate the most common errors made in writing loops, and how to detect and correct these errors.

Lesson 23, Exercise 4:

```
54.1  SET A = 1
54.2  TYPE "CONVERSION OF FEET TO INCHES"
54.3  TYPE A, 12*A IN FORM 2
54.4  SET A = A + 1
54.5  TO STEP 54.3
FORM 2:
←←←FEET = ←←←←INCHES
```

WHICH STEP IS WRONG?

Lesson 23, Exercise 5:

```
56.1  LET F(X) = 3*X↑3 + 5
56.2  TYPE X, F(X)
56.3  SET X = X + .5
56.4  TO STEP 56.2 IF X < 3.5
```

IN PART 56 THERE IS A MISSING STEP. IT SHOULD GO BEFORE

WHAT STEP?

Lesson 24: Loops with variables in the exit condition. In Lesson 24 the study of loops is continued and extended to include loops in which the counter is compared not to a constant but to another variable. The use of variables for initial values of the counter is also discussed.

Lesson 24, Exercise 12:

IN A LOOP BOTH THE INITIAL VALUE AND THE VALUE USED FOR COMPARISON IN THE EXIT CONDITION CAN BE VARIABLES. THE VALUES OF THE VARIABLES WILL BE GIVEN BY THE USER BEFORE THE LOOP IS USED.

```
6.51  DEMAND L
6.52  DEMAND U
6.53  SET X = L
6.54  TYPE X*3/17
6.55  SET X = X + 1
6.56  TO STEP 6.54 IF X < U
```

WHAT VARIABLE IS USED FOR THE INITIAL VALUE?

Lesson 24, Exercise 13:

HERE IS A MORE COMPLICATED PROGRAM. TRY TO FIND OUT WHAT THE PROGRAM DOES.

```
3.1  SET N = 1
3.2  SET S = 0
3.3  TYPE N
3.4  SET S = S + N
3.5  SET N = N + 1
3.6  TO STEP 3.3 IF N < 4
3.7  TYPE S
```

Lesson 25: Loops and the FOR clause. In Lesson 25 loops are reviewed and practiced. The students are taught that many programs with loops are equivalent to simpler programs that are executed iteratively by using a DO command with a FOR clause; in several exercises the students are asked to rewrite looping programs so as to be able to use the FOR clause, and are advised to use this simpler approach because of the greater ease in writing and debugging such programs. Students are cautioned that not every looping program can be rewritten in this way, and examples are shown to illustrate the point.

Lesson 25, Exercise 7:

```
4.1  SET C = 1
4.2  TYPE 60/C
4.3  SET C = C + 1
4.4  TO STEP 4.2 IF C < 7
```

WHAT DOES THIS PROGRAM DO?

Lesson 25, Exercise 7.4:

HERE IS THE ABOVE PROGRAM REWRITTEN TO USE A "FOR" CLAUSE:

```
4.1  TYPE 60/C
DO STEP 4.1 FOR ...
```

COMPLETE THE "FOR" CLAUSE.

Lesson 26: Loops with a DEMAND command. In interactive programming it is a frequent practice to allow for input from the user during the execution of a loop; in AID this is done with a DEMAND command. When a DEMAND command is executed and the response from the user is empty, i.e., a carriage return only, AID terminates the execution of the program at that point. Because of this feature, loops that incorporate DEMAND commands need no conditional clause to determine when to cease iteration.

Lesson 26, Exercise 2:

IF THE USER TYPED THE RETURN KEY ONLY, INSTEAD OF TYPING A VALUE FOR L, AID WOULD STOP THE PROGRAM (THIS IS LIKE AN EXIT CONDITION).

```
5.1  DEMAND R
5.2  SET A = 3.14159265*R↑2
5.3  TYPE A
```

5.4 TO STEP 5.1

WHAT DOES PART 5 DO?

Lesson 26, Exercise 5:

WRITE A PROGRAM TO CONVERT INCHES TO FEET AND INCHES.
USE A "DEMAND" COMMAND IN A LOOP. START AID AND TEST
THESE VALUES:

159 INCHES

17 INCHES

44 INCHES

Lesson 27: Test of Lessons 22 to 26. This lesson, the block test for the fourth lesson block, contains 30 exercises, of which 29 are test exercises. The test exercises are classified by type in Table 3, and are classified in Table 4 by the lesson being tested.

Lesson 28: Block review. Lesson 28 is the block review for the fourth lesson block, and like the block review in Lesson 21 refers students to particular parts of the teaching lessons for review of specific subjects. In addition, the contents of the three previous lesson blocks are summarized and the students are referred to the appropriate block reviews (Lessons 7, 14, and 21) for review and practice on the topics covered in preceding blocks.

Lesson 29: Absolute values. Lesson 29 reviews the concept of absolute value of real numbers and introduces the AID notation for absolute value: !x! . The hierarchy of arithmetic operations is reviewed and the place of absolute value in the hierarchy is specified. The use of absolute values for finding Euclidean distances is discussed.

Lesson 29, Exercise 17:

!17.1 - 7.9! IS THE DISTANCE BETWEEN 7.1 AND WHAT?

Lesson 29, Exercise 19:

WRITE A PROGRAM TO FIND WHICH OF THE THREE NUMBERS A, B, AND C IS CLOSEST TO 13/17. HAVE YOUR PROGRAM PRINT ONE OF THESE MESSAGES:

A IS CLOSEST TO 13/17.

B IS CLOSEST TO 13/17.

C IS CLOSEST TO 13/17.

Lesson 30: SIN and COS. Lesson 30 is an optional lesson introducing the basic trigonometric functions for students who are interested and have some background in the subject. Some basic trigonometric notions are reviewed, the two standard AID functions are introduced, and there are a few practice problems involving the definitions of the tangent and secant functions. None of the problems are of great difficulty and the lesson is very short (17 exercises).

Lesson 30, Exercise 9:

DEFINE A TANGENT FUNCTION, T(X), WHICH WILL FIND THE TANGENT WHEN X IS GIVEN IN RADIANS. START THE AID INTERPRETER AND USE YOUR FUNCTION TO FIND THE TANGENT WHEN X = 0, 2.4, 3.1, -6.

Lesson 31: Exponential and logarithmic functions. Lesson 31 is also optional and provides a brief introduction to the exponential and logarithmic functions that can be used in AID programming; this lesson is intended only for students with the appropriate background and interest. The concepts of exponential and logarithmic functions are

reviewed and the AID notation for e^x and $\ell n(x)$ is introduced. There are a few problems on the conversion of bases, although the treatment of this topic is adequate only for students who have already studied it.

Lesson 31, Exercise 10:

> USING AID, FIND THE VALUES OF EXP(X) FOR X = .5, 1.0, 1.5,...,10.0

Lesson 31, Exercise 12:

> DEFINE A FUNCTION T(X) WHICH WILL COMPUTE THE LOGARITHM TO THE BASE 2 OF X. TEST THE FUNCTION FOR THESE VALUES OF X:
>
> .6, 5, 8.7, 100

Lesson 32: Lists. The first three lessons in this lesson block were relatively short and easy, and were devoted more to mathematical concepts than programming concepts. Lesson 32, in contrast, is quite long and very difficult, introducing one of the most complex topics in programming: the storage and use of indexed arrays.

Lesson 32 begins with a discussion of indices and the AID notation for them. A simple program for inputting a list of numbers is shown and several difficult programming problems whose solution depends upon the retrieval of data from stored lists follow.

Lesson 32, Exercise 8:

> WRITE A PROGRAM TO FIND THE AVERAGE OF THE NUMBERS IN A LIST OF TEN NUMBERS.

Lesson 32, Exercise 19:

> WRITE A PROGRAM TO FIND AND PRINT ALL THE NUMBERS LESS THAN 30 IN A LIST OF 10 NUMBERS.

Lesson 33: Using loops with lists of numbers. Lesson 33 continues the treatment of stored arrays introduced in Lesson 32. The first topic covered in Lesson 33 is a method for counting and simultaneously storing the elements in a list so that later computations, such as finding the average, can be done more easily. Students are shown several examples of looping programs that use data stored in lists, and are asked to write five programs of this kind.

Lesson 33, Exercise 3:

```
WHAT IS THE PURPOSE OF THIS PROGRAM?

    2.1  SET I = 1
    2.2  TYPE L(I) IF L(I) # 0
    2.3  SET I = I + 1
    2.4  TO STEP 2.2 IF I <= N
```

Lesson 33, Exercise 6:

```
SUPPOSE L IS THE LIST 12, 0, -7, 0, 0, 8.  (N IS THE
LENGTH OF L)  HERE IS A PROGRAM THAT BUILDS A NEW LIST A.

WHAT IS THE VALUE OF A(1) AFTER THIS PROGRAM IS RUN?

    7.1  SET I = 1
    7.2  SET A(I) = 1 IF L(I) # 0
    7.3  SET A(I) = 0 IF L(I) = 0
    7.4  SET I = I + 1
    7.5  TO STEP 7.2 IF I <= N
```

Lesson 33 is not very long (23 exercises) but there is a rather high proportion of programming problems (5) of medium difficulty.

Lesson 34: Test of Lessons 29 to 33. Lesson 34 is the test for the fifth lesson block. There are 26 test exercises and one request for student opinions in the lessons. The 26 test exercises are classified in Table 4 by the lesson they are testing.

Since Lessons 30 and 31 were optional, the corresponding exercises in the test (Exercises 8 through 17) are also optional.

Lesson 35: Block review. Lesson 35 is the block review for the fifth lesson block. About half of the lesson contains review exercises and the other half gives students references to exercises that cover specific topics.

Lesson 36: Nested loops and decrementing counters. This lesson continues work on loops, and starts by showing several examples of programs in which the conditional clause that determines the number of iterations is formed by comparing the value of the counter to the value of an algebraic expression (rather than to a simple variable or constant). Nested loops are introduced, and decremented counters are used in several problems.

Lesson 36, Exercise 10:

> WRITE A PROGRAM USING NESTED LOOPS WHICH WILL MAKE ONE TABLE OF INTEREST CORRESPONDING TO A RATE OF 6% AND THE FOLLOWING GROUPS OF PRINCIPLES:
>
> 50, 100, 150, 200; 250, 500, 750, 1000; 1250, 2500, 3250, 5000.

Lesson 37: SUM, PROD, MAX, and MIN. Four AID functions that are very useful for finding the sum, product, maximum or minimum of a sequence of numbers are introduced in this lesson. The sequence of numbers may be expressed by simply listing them as the argument of the function:

SUM(4.53, 3.72, 6.29, 7.81)

or by a formula:

SUM(J = 1, 2, 3, 4 : J*3)

or by giving the variable name of a stored list:

SUM(J = 1, 2, 3, 4 : L(J))

When the sequence is defined by formula or is to be found in a stored list, the indices to be used must be specified either by a simple listing of the indices or by a "range specification."

The lesson starts with a brief review of numeric sequences and formulas for sequences before the syntax of the new AID functions is introduced.

Lesson 37, Exercise 18:

USE AID TO FIND THE PRODUCT OF

1, 4, 9, ..., 100

Lesson 37, Exercise 26:

WHAT WILL AID ANSWER?

```
SET X = 24
SET Y = X/3
TYPE MIN(SUM(X,Y), PROD(X,Y))
```

Lesson 38: Arrays. In Lesson 38 two dimensional arrays are introduced. After some discussion of ordered indices, the lesson gives a simple program for storing data in two-dimensional arrays and then gives the student several programming problems using data stored in this form.

Lesson 38, Exercise 8:

STORE THIS TABLE AS A 5 BY 3 ARRAY A.

1	5	25
2	10	100
3	15	225
4	20	400
5	25	625

USE "TYPE A" TO GET A LISTING OF YOUR ARRAY.

Lesson 38, Exercise 9:

WRITE A PROGRAM THAT WILL ADD THE COLUMNS OF A 5 BY 3 ARRAY. THE PROGRAM SHOULD TYPE

THE SUM OF COLUMN 1 IS ...

THE SUM OF COLUMN 2 IS ...

THE SUM OF COLUMN 3 IS ...

Lesson 39: More about arrays and lists. Lesson 39 discusses the limitations on the dimension of AID arrays and the permissible range of the indices. The use of algebraic expressions as indices is also mentioned. Several quite difficult programming problems are given, as illustrated in the example below.

Lesson 39, Exercise 8:

STORE THE FOLLOWING ARRAYS:

A: 3 BY 4 WHERE A(I,J) = 3*I - J

B: 3 BY 4 WHERE B(I,J) = -2*I + J

THEN FORM A NEW 3 BY 4 ARRAY, E, WHOSE ELEMENT IN THE ITH ROW AND JTH COLUMN IS THE MAXIMUM OF THE ELEMENTS IN THE SAME POSITION IN THE ARRAYS A AND B.

Lesson 40: Conditional definition of functions. One important feature of AID is the simplicity of its conditional definition of functions. Since the conditional definition of functions depends upon the use of Boolean expressions, the lesson begins with a brief survey of these expressions. The syntax of conditional definitions is then given and a number of examples are presented.

In mathematics, we often encounter functions that cannot be defined by a single formula but may, instead, be defined by several formulas each applying to a particular part of the domain of the function; the definitions of such functions usually are given in a form such as

If CONDITION 1 then f(x) = EXPRESSION 1

If CONDITION 2 then f(x) = EXPRESSION 2

etc.

In AID this definition is expressed as follows:

LET F(X) = (CONDITION 1: EXPRESSION 1; CONDITION 2: EXPRESSION 2; ...)

In some cases a function may be defined like this:

If CONDITION then f(x) = EXPRESSION 1

Otherwise f(x) = EXPRESSION 2.

This kind of definition is written in AID by simply omitting the final condition and using the final expression as the definition of the function for all cases where one of the preceding conditions does not hold:

LET F(X) = (CONDITION: EXPRESSION 1; EXPRESSION 2)

Lesson 40, Exercise 13:

WRITE THE CONDITIONAL DEFINITION OF A FUNCTION F(X) SUCH THAT

IF X < 0 THEN F(X) = X↑2

IF X >= 0 THEN F(X) = X↑3

Lesson 40, Exercise 21:

LOCAL FIRST CLASS POSTAL RATES UP TO 32 OUNCES ARE DEFINED AS FOLLOWS, WHERE W IS IN OUNCES:

IF W <= 13, THE COST IS $.06 PER WHOLE OUNCE, PLUS $.06 FOR ANY FRACTION OF AN OUNCE.

IF 13 < W <= 16, THE COST IS $.80

IF 16 < W <= 24, THE COST IS $.98

IF 24 < W <= 32, THE COST IS $1.16.

WRITE AND RUN A PROGRAM TO COMPUTE POSTAGE COSTS.

Lesson 41: Test of Lessons 36 to 40. Lesson 41, like other tests, allows the students only one trial per exercise. If a student cannot answer a question, he can request the correct answer and go to the next exercise. There are no hints provided by the program. The number of exercises of each kind are listed in Table 3, and are also classified in Table 4 according to which teaching lesson is being tested.

As before, test exercises may test more than one lesson but each exercise is listed only once in Table 4.

Lesson 42: Block review. Lesson 42 is the block review for the sixth lesson block, covering the same lessons as the test in the preceding Lesson 41. Here again students who want to review a particular topic are referred to appropriate exercises in the teaching lessons.

Lesson 43: Recursive functions. Recursive functions are defined in the same form as other conditional functions and are presented only for those students with appropriate background and interests. The most commonly used examples of recursive functions, such as the factorial and the Fibonacci numbers, are used as examples. Since this lesson is intended only for those students who are somewhat more sophisticated mathematically, the exercises are correspondingly more difficult than other lessons in the course.

Lesson 43, Exercise 6:

WRITE A PROGRAM CONTAINING A RECURSIVE FUNCTION N(A,X,E)
THAT USES NEWTON'S ALGORITHM FOR OBTAINING THE APPROXIMATE
SQUARE ROOT OF THE NUMBER A.

A = POSITIVE NUMBER WHOSE SQUARE ROOT WILL BE APPROXIMATED

X = FIRST APPROXIMATION TO SQRT(A)

E = ALLOWABLE DIFFERENCE BETWEEN X↑2 AND A

Lesson 44: AND, OR, and NOT; truth tables. The Boolean connectives AND, OR, and NOT are discussed in detail in this lesson and the hierarchy is given explicitly. Truth tables are introduced and the truth tables for various compound statements are used in the exercises. This lesson is primarily an introduction to sentential logic for students who have not previously studied the subject, and the exercises also serve as vehicles for further practice using AID.

Lesson 44, Exercise 10:

WHICH OF THE FOLLOWING STATEMENTS ARE TRUE?

A. -1 < 2 OR 3 > 4

B. -1 < 2 AND 3 > 4

C. 6 = 7 OR 9 < 1

D. .5 # 0 OR 0 < .5

E. 9 >= 11 OR (6 > 1 AND 1 < 2)

Lesson 45: TV(X) and the FIRST function. TV and FIRST are two advanced AID functions that are useful in special circumstances. TV is used only for Boolean expressions and takes on the values 0 (false) and 1 (true). The function FIRST is used to find the first number in a sequence that satisfies a given condition; for example, FIRST(K = 6(2)14 : (K/2)↑2 > 24) will give the first number K in the sequence 6, 8, 10, 12, 14 such that (K/2)↑2 is greater than 24.

Lesson 45, Exercise 5:

HOW WOULD YOU DEFINE A FUNCTION F SUCH THAT

F(X) = 1 IF X IS FALSE

F(X) = 0 IF X IS TRUE

Lesson 45, Exercise 16:

GIVEN THE SEQUENCE

0, .2, .4, .6, ..., Z*.2, ...

WRITE A PROGRAM THAT WILL FIND THE FIRST MEMBER OF THIS SEQUENCE SUCH THAT SIN(Z*.2) < 0 AND THE FIRST MEMBER AFTER THIS ONE.

Lesson 46: LET and Boolean expressions; debugging tools. The first topic covered in Lesson 46 is the use of LET to assign a variable name to a Boolean expression. For example, to assign to the variable S the sentence "P or not Q or not R", the following command is used:

```
LET S = P OR NOT Q OR NOT R
```

P, Q, and R must, of course, be given Boolean values before S is called.

The remainder of Lesson 46 is devoted to several useful debugging tools. The first of these is the GO command which requests AID to continue executing an interrupted program. The GO command is used when the execution of a program is interrupted because of an error; after correcting the error, the programmer may type "GO" to continue. Another command that helps to debug complex programs is DONE. DONE is used indirectly, i.e., as part of the stored program, and its effect is to cause the execution of the program to stop at that point. DONE is most often used conditionally to stop execution under certain conditions, for example, when a variable assumes a value that is out of bounds:

```
6.58  DONE IF X > 10↑8
```

Lesson 46, Exercise 3:

```
WHAT VALUE WILL AID TYPE FOR A?

     LET A = B AND C

     SET B = TRUE

     SET C = FALSE

     TYPE A
```

Lesson 46, Exercise 14:

THE "DONE" COMMAND HAS A PARTICULAR APPLICATION FOR LOOPS. SOMETIMES YOU MIGHT WANT TO TEST A LOOP FOR ONLY ONE OR TWO LOOPINGS. THE CONDITIONAL DONE COMMAND CAN BE USED FOR THIS.

```
5.1  SET N = 1
5.2  SET K = 1/N
5.3  TYPE N,K
5.4  SET N = N + 1
5.5  TO STEP 5.2 IF N <= 100
```

WHICH OF THE FOLLOWING COULD BE USED TO STOP THE LOOPING AFTER TWO LOOPS?

A. 5.42 DONE IF K = 1/4

B. 5.21 DONE IF N = 5

C. 5.36 DONE IF N = 2

D. 5.6 DONE IF N = 2

N. NONE OF THE ABOVE

Lesson 47: More standard AID functions. The two AID functions that are covered in this lesson are DP (digit part) and XP (exponent part). The lesson starts with a review of scientific notation, explaining that two parts of a number in scientific notation are the digit part and the exponent part. The digit part of a number can be found by using the digit part function, DP(X), and the exponent part by using the exponent part function, XP(X).

Lesson 47, Exercise 4:

WHAT IS THE EXPONENT PART OF 8325.6?

Lesson 47, Exercise 12:

WRITE A PROGRAM THAT WILL TAKE ANY NUMBER X AND ROUND IT TO THREE SIGNIFICANT DIGITS.

Lesson 48: Test of Lessons 43 to 47. Lesson 48 is the test for the seventh, and last, lesson block in the course. Of the 32 exercises in the lesson, two are requests for students' opinions. The other 30 exercises test one of Lessons 43 to 47, as shown in Table 4.

Lesson 48 has a rather high proportion of true-false exercises (7) and no programming problems.

Lesson 49: Block review. Lesson 49 is the review of the seventh lesson block and covers Lessons 43 to 47. The contents of each of the teaching lessons in the block are summarized briefly, and students are asked if they want to review any part of the lessons. If a student wants to review a particular topic, he is given a reference to pertinent exercises.

Lesson 50: Programming problems. Lesson 50 is not part of a lesson block, nor is it a teaching lesson, but rather a collection of lengthy and difficult programming problems for students who want to practice the programming skills they have acquired in the course. Some of the exercises require considerable mathematical knowledge and sophistication; these exercises are intended only for the student with an appropriate background. Other problems, though difficult, are accessible to students with less mathematical background.

APPENDIX

Table of Contents: Teaching Lessons

	Number of Exercises	List of Exercises
Lesson 1: Using the Instructional Program		
How to answer	4	1, 2, 3, 4
How to erase	3	5, 6, 7
How to get hints and answers	2	8, 9
How to use Ctrl-G	4	10, 11, 11.1, 12
How to sign on and off	3	13, 13.1, 13.2
Prompted decisions*	2	14, 15
Lesson 2: Using AID for Arithmetic		
The AID interpreter	4	1, 2, 17, 18
Symbols for arithmetic operations	9	8, 9, 10, 11, 12, 13, 14, 14.1, 14.2
The TYPE command	12	3, 4, 5, 6, 7, 15, 16, 16.1, 16.2, 19, 19.1, 19.2
Prompted decisions	3	20, 21, 21.1
Lesson 3: Order of Arithmetic Operations		
Use of parentheses	22	1, 2, 2.1, 3, 4, 4.1 to 4.6, 5, 5.1, 6, 7, 8, 8.1, 8.2, 9, 20, 20.1, 20.2
Hierarchy of operations	21	10, 10.1, 10.2, 11, 11.1, 12, 12.1, 12.2, 13, 13.1, 13.2, 14, 15, 16, 21, 22, 23, 24, 24.1 to 24.3
Negative numbers	3	17, 18, 19
Prompted decisions	3	25, 26, 27
Lesson 4: Exponents and Scientific Notation		
Exponents	12	1, 1.1, 2, 2.1, 2.2, 2.3, 5, 5.1, 22, 22.1 to 22.3
Using zero and one as exponents	2	3, 4

*Exercises that ask the student to state a preference for the sequence of instruction.

	Number of Exercises	List of Exercises
Order of operations	20	6, 6.1 to 6.8, 7, 8, 9, 9.1, 10, 10.1, 11, 12, 12.1 to 12.3
Using fractional exponents	3	13, 14, 15
Negative exponents	9	16, 17, 17.1, 17.2, 18, 19, 20, 20.1, 21
Reading scientific notation	7	23, 23.1, 23.2, 25, 25.1, 27, 27.1
Writing scientific notation	5	24, 24.1, 26, 26.1, 26.2
Prompted decisions	3	28, 29, 30
Lesson 5: The SET and DELETE Commands		
The SET command	47	1, 2, 3, 4, 5, 5.1, 6, 7, 7.1, 7.2, 8, 9, 10, 11, 11.1, 11.2, 12, 12.1, 13, 13.1, 13.2, 14, 14.1, 15, 15.1, 16, 16.1, 17, 17.1, 18, 18.1, 18.2, 19, 19.1, 19.2, 20, 20.1, 21, 21.1, 21.2, 30, 30.1, 30.2, 30.3, 31, 31.1, 31.2
The DELETE command	6	22, 23, 23.1, 24, 25, 26
The multiple TYPE command	7	27, 27.1, 28, 29, 29.1 to 29.3
Prompted decisions	3	32, 33, 34
Lesson 8: The LET Command		
Functions of one variable	30	1, 1.1 to 1.3, 2, 2.1, 2.2, 3, 4, 5, 6, 7, 8, 8.1, 9, 9.1, 10, 10.1 to 10.3, 22, 23, 23.1, 28, 28.1, 29, 30, 30.1, 30.2, 31
Functions of two or more variables	15	11, 11.1 to 11.5, 12, 13, 14, 15, 16, 17, 27, 27.1, 27.2
Substituting algebraic expressions for variables	11	18, 19, 20, 20.1 to 20.4, 21, 24, 24.1, 24.2
Printing and deleting definitions of functions	2	25, 26
Prompted decisions	4	32, 33, 34, 35

	Number of Exercises	List of Exercises
Lesson 9: Some Standard AID Functions		
The SQRT function	5	1, 2, 3, 3.1, 3.2
The IP function	7	4, 5, 6, 7, 8, 8.1, 8.2
The FP function	9	9, 10, 11, 12, 13, 14, 14.1, 14.2, 15
The SGN function	6	16, 17, 18, 19, 20, 21
Prompted decisions	4	22, 23, 24, 25
Lesson 10: Indirect Steps, the DO Command, the FOR Clause		
Step numbers and the DO command	12	1, 1.1, 2, 3, 4, 5, 5.1, 6, 6.1, 7, 12, 12.1
Deleting, replacing, and printing indirect steps	3	8, 9, 10
The FOR clause	2	11, 11.1
Range specifications	13	13, 13.1, 14, 15, 15.1, 15.2, 16, 16.1, 16.2, 17, 18, 19, 19.1
Prompted decisions	3	20, 21, 22
Lesson 11: Parts		
Parts	2	1, 2
DO PART ...	3	3, 4, 4.1
DO PART ... FOR ...	5	8, 9, 10, 11, 11.1
Printing and deleting parts	6	12, 12.1, 13, 13.1, 14, 14.1
Sequence of execution	10	5, 5.1, 6, 7, 7.1, 15, 15.1 to 15.4
Prompted decisions	4	16, 17, 18, 19
Lesson 12: The DEMAND Command		
The DEMAND command	7	1, 1.1, 2, 3, 3.1, 4, 4.1
Answering a DEMAND with a "return"	2	7, 7.1
DO PART ..., ... TIMES	2	5, 6
Prompted decisions	3	8, 9, 10
Lesson 15: Relations and the Use of the IF Clause		
Relation symbols	23	1, 1.1, 1.2, 3, 3.1 to 3.4, 4, 4.1 to 4.3, 5, 5.1 to 5.4, 6, 6.1 to 6.5

	Number of Exercises	List of Exercises
The number line	8	2, 2.1 to 2.7
Positive and negative	6	7, 8, 9, 9.1, 9.2, 10
The IF clause	14	11, 12, 12.1, 13, 13.1, 14, 14.1, 14.2, 15, 15.1, 16, 17, 17.1, 18
Using AND and OR in IF clauses	7	19, 19.1, 20, 20.1, 21, 21.1, 21.2
Prompted decisions	4	22, 23, 24, 25
Lesson 16: The TO Command		
The TO command	7	4, 4.1, 4.2, 5, 6, 6.1, 6.2
Endless loops	7	2, 2.1 to 2.6
Sequence of execution	10	1, 1.1 to 1.4, 3, 3.1 to 3.4
Prompted decisions	3	7, 8, 9
Lesson 17: Debugging Techniques		
Tracing values of variables	12	1, 1.1, 1.2, 2, 2.1 to 2.4, 3, 3.1 to 3.3
Sequence of execution	1	7
Tracing expected output	7	4, 5, 5.1 to 5.5
Writing a complete trace	4	6, 7.1, 7.2, 7.3
Prompted decisions	3	8, 9, 10
Lesson 18: The Indirect Use of the DO Command		
The indirect use of DO	25	1.1 to 1.20, 4, 4.1, 5, 6, 7
Sequence of execution	6	2, 2.1, 2.2, 3, 3.1, 3.2
Prompted decisions	3	8, 9, 10
Lesson 19: Debugging, Permanent Storage		
Planning a program	6	2, 2.1 to 2.4, 3
Editing the program	5	4, 4.1, 5, 5.1, 8
Testing the program	2	6, 11
Syntax and semantic errors	3	7, 7.1, 7.2
Executing the program step-by-step	3	9, 9.1, 10
Disk storage	9	12, 13, 14, 15, 15.1, 16, 17, 18, 19
Prompted decisions	4	1, 1.1, 20, 21

	Number of Exercises	List of Exercises
Lesson 22: The FORM Statement		
The FORM statement	13	1, 2, 3, 4, 5, 6, 7, 8, 10, 10.1, 10.2, 11, 12
Rounding	3	9, 9.1, 12.1
TYPE "..."	2	17, 17.2
TYPE ←	3	18, 19, 19.1
Using a FORM statement for more than one number	5	13, 14, 15, 16, 16.1
Prompted decisions	3	20, 21, 22
Lesson 23: Loops		
Loops	33	1, 1.1, 1.2, 2, 2.1 to 2.14, 3, 3.1, 3.2, 4, 4.1, 4.2, 5, 5.1, 6, 6.1, 6.2, 7, 7.1, 8, 8.1
Prompted decisions	3	9, 10, 11
Lesson 24: Loops with Variables in the Exit Condition		
Using a variable in the exit condition	16	1, 1.1 to 1.3, 2, 2.1 to 2.4, 3, 4, 4.1, 4.2, 5, 10, 11
Using a variable for the initial value	3	12, 12.1, 12.2
Common errors in loops	4	6, 7, 8, 9
Other ways to use loops	9	13, 13.1 to 13.8
Prompted decisions	3	14, 15, 16
Lesson 25: Loops and the FOR Clause		
Replacing a loop with DO PART ... FOR ...	17	1, 2, 3, 4, 5, 6, 6.1 to 6.4, 7, 7.1 to 7.4, 8, 8.1
Prompted decisions	3	9, 10, 11
Lesson 26: Loops with a DEMAND Command		
Loops with a DEMAND command	16	1, 2, 3, 3.1, 4, 4.1 to 4.3, 5, 5.1, 6, 6.1, 6.2, 7, 8, 8.1
Prompted decisions	3	9, 10, 11

	Number of Exercises	List of Exercises
Lesson 29: Absolute Value		
Absolute value	14	1, 1.1, 2, 2.1 to 2.3, 3, 3.1, 4, 5, 6, 15, 15.1, 15.2
Hierarchy of operations	8	7, 8, 9, 10, 11, 12, 13, 14
Distance	8	16, 17, 18, 18.1 to 18.3, 19, 19.1
Prompted decisions	3	20, 20.1, 21
Lesson 30: SIN and COS		
SIN and COS	10	1, 2, 3, 4, 5, 5.1, 5.2, 6, 6.1, 10
Radians and degrees	2	7, 7.1
Other trigonometric functions	3	8, 9, 9.1
Prompted decisions	2	11, 12
Lesson 31: Exponential and Logarithmic Functions		
Exponents and bases	8	1, 1.1, 2, 2.1, 3, 3.1, 4, 4.1
Exponential functions	8	5, 6, 6.1 to 6.3, 7, 8, 8.1
EXP(X)	3	9, 10, 10.1
LOG(X)	3	11, 12, 12.1
Prompted decisions	2	13, 13.1
Lesson 32: Lists		
Lists and indices	16	1, 1.1 to 1.4, 2, 2.1, 2.2, 3, 4, 4.1 to 4.3, 16, 17, 18
Programs that use lists	25	5, 5.1, 6, 6.1 to 6.8, 7, 7.1, 7.2, 8, 8.1, 9, 10, 11, 11.1, 11.2, 19, 19.1 to 19.3
LET S BE SPARSE	4	12, 13, 13.1, 13.2
Printing a list	2	14, 15
Prompted decisions	3	20, 21, 22
Lesson 33: Using Loops with Lists of Numbers		
Using loops with lists	14	1, 2, 2.1, 3, 3.1, 4, 4.1, 5, 5.1, 5.2, 7, 7.1, 8, 8.1
Using loops to make new lists	6	6, 6.1 to 6.3, 9, 9.1
Prompted decisions	3	10, 11, 12

	Number of Exercises	List of Exercises
Lesson 36: Nested Loops and Decrementing Counters		
Using algebraic expressions in the exit condition	9	1, 1.1, 2, 2.1, 3, 3.1, 3.2, 4, 4.1
Nested loops	29	5, 5.1 to 5.4, 6, 6.1 to 6.6, 7, 7.1 to 7.6, 8, 8.1 to 8.4, 9, 9.1, 9.2, 10, 10.1
Decrementing counters	17	11, 11.1, 11.2, 12, 12.1 to 12.3, 13, 13.1, 13.2, 14, 14.1 to 14.3, 15.1, 16, 16.1
Prompted decisions	3	17, 18, 19
Lesson 37: SUM, PROD, MAX, and MIN		
Sequences and formulas	4	1, 2, 3, 4
SUM used with formulas	9	5, 6, 7, 8, 9, 10, 11, 12, 12.1
SUM used with lists	10	13, 14, 15, 20, 20.1 to 20.4, 27, 27.1
PROD	5	16, 17, 18, 18.1, 19
MAX	4	21, 23, 25, 25.1
MIN	5	22, 24, 25.2, 28, 28.1
Using SUM, PROD, MAX and MIN with listed arguments	4	26, 26.1 to 26.3
Prompted decisions	2	29, 30
Lesson 38: Arrays		
Arrays	4	1, 1.1, 2, 3
Storing arrays	14	4, 4.1, 5, 6, 6.1 to 6.4, 7, 7.1, 7.2, 8, 10, 10.1
Using arrays	7	6.5, 6.6, 6.7, 9, 9.1, 10.2, 10.3
LET A BE SPARSE	6	11, 11.1, 12, 12.1 to 12.3
Prompted decisions	3	13, 14, 15
Lesson 39: More about Arrays and Lists		
Subscripts for arrays	12	1, 1.1, 2, 2.1, 2.2, 9, 10, 10.1 to 10.3, 11, 11.1
Nested DO commands used to store arrays	13	3, 3.1, 4, 5, 5.1, 5.2, 6, 6.1, 6.2, 7, 7.1, 8, 8.1

	Number of Exercises	List of Exercises
Using arrays	9	12, 12.1 to 12.3, 13, 14, 14.1 to 14.3
Prompted decisions	3	15, 16, 17
Lesson 40: Conditional Definition of Functions		
Boolean expressions	6	1, 2, 3, 4, 5, 6
Conditional expressions	6	7, 8, 9, 10, 11, 11.1
Conditional definition of functions	13	12, 13, 13.1 to 13.3, 16, 17, 18, 18.1, 21, 22, 23, 24
Terminating clause in conditional definition of functions	2	14, 15
Ordering of clauses	2	19, 20
Prompted decisions	2	25, 26
Lesson 43: Recursive Functions		
Recursive functions	22	1, 2, 2.1, 3, 3.1 to 3.4, 4, 4.1, 5, 6, 6.1, 6.2, 7, 7.1 to 7.3, 8, 8.1 to 8.3
Prompted decisions	2	9, 10
Lesson 44: AND, OR, and NOT; Truth Tables		
AND, OR, and NOT	2	1, 2
Truth value of Boolean expressions	5	3, 4, 5, 6, 7
Conjunctions	4	8, 8.1, 8.2, 9
Disjunctions	2	10, 11
Order of evaluation	6	12, 13, 14, 15, 15.1, 15.2
Truth tables	4	16, 17, 18, 19
AND chains	3	20, 21, 22
Prompted decisions	2	23, 24
Lesson 45: TV(X) and the FIRST Function		
TV(X)	9	1, 1.1, 2, 3, 4, 4.1, 5, 6, 7
The function FIRST	11	8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 17.1
Prompted decisions	3	18, 19, 20

	Number of Exercises	List of Exercises
Lesson 46: LET and Boolean Expressions; Debugging Tools		
Using LET with Boolean expressions	6	1, 2, 3, 4, 5, 6
GO	5	7, 7.1, 8, 9, 9.1
DONE	9	10, 11, 1.1, 12, 13, 14, 15, 16, 16.1
Prompted decisions	3	17, 18, 19
Lesson 47: More Standard AID Functions		
DP(X)	11	1, 2, 3, 4, 5, 5.1, 7, 7.2, 10, 11, 12
XP(X)	7	4.1, 6, 6.1, 7.1, 7.3, 8, 9
Prompted decisions	3	13, 14, 15

References

Friend, J. Student manual, introduction to programming: AID. Institute for Mathematical Studies in the Social Sciences, Stanford University, 1969.

Friend, J., & Atkinson, R. C. Computer-assisted instruction in programming: AID. Technical Report No. 164, Institute for Mathematical Studies in the Social Sciences, Stanford University, January 25, 1971.

Friend, J. INSTRUCT coders' manual. Technical Report No. 172, Institute for Mathematical Studies in the Social Sciences, Stanford University, May 1, 1971.

Suppes, P., Goldberg, A., Kanz, G., Searle, B., & Stauffer, C. Teacher's handbook for CAI courses. Technical Report No. 178, Institute for Mathematical Studies in the Social Sciences, Stanford University, September 1, 1971.

Friend, J. E., Fletcher, J. D., & Atkinson, R. C. Student performance in computer-assisted instruction in programming. Technical Report No. 184, Institute for Mathematical Studies in the Social Sciences, Stanford University, May 10, 1972.

Footnotes

[1]Funding support provided by ONR Grant N00014-67-A-0112-0054.

[2]All excerpts quoted from the AID course are given in upper case characters, in imitation of the form in which they appear on student teletypewriter terminals.

DISTRIBUTION LIST

Navy

4 Dr. Marshall J. Farr, Director
Personnel & Training Research Programs
Office of Naval Research
Arlington, VA 22217

1 Director
ONR Branch Office
495 Summer Street
Boston, MA 02210
ATTN: C. M. Harsh

1 Director
ONR Branch Office
1030 East Green Street
Pasadena, CA 91101
ATTN: E. E. Gloye

1 Director
ONR Branch Office
536 South Clark Street
Chicago, IL 60605
ATTN: M. A. Bertin

1 Office of Naval Research
Area Office
207 West 24th Street
New York, NY 10011

6 Director
Naval Research Laboratory
Code 2627
Washington, DC 20390

12 Defense Documentation Center
Cameron Station, Building 5
5010 Duke Street
Alexandria, VA 22314

1 Chairman
Behavioral Science Department
Naval Command and Management Division
U.S. Naval Academy
Luce Hall
Annapolis, MD 21402

1 Chief of Naval Technical Training
Naval Air Station Memphis (75)
Millington, TN 38054
ATTN: Dr. G. D. Mayo

1 Chief of Naval Training
Naval Air Station
Pensacola, FL 32508
ATTN: Capt. Allen E. McMichael

1 LCDR Charles J. Theisen, Jr., MSC, USN
4024
Naval Air Development Center
Warminster, PA 18974

1 Commander
Naval Air Reserve
Naval Air Station
Glenview, IL 60026

1 Commander
Naval Air Systems Command
Department of the Navy
AIR-413C
Washington, DC 20360

1 Mr. Lee Miller (AIR-413E)
Naval Air Systems Command
5600 Columbia Pike
Falls Church, VA 22042

1 Dr. Harold Booher
NAVAIR 415C
Naval Air Systems Command
5600 Columbia Pike
Falls Church, VA 22042

1 Capt. John F. Riley, USN
Commanding Officer
U.S. Naval Amphibious School
Coronado, CA 92155

1 Special Assistant for Manpower
OASN (M&RA)
The Pentagon, Room 4E794
Washington, DC 20350

1 Dr. Richard J. Niehaus
Office of Civilian Manpower Management
Code 06A
Department of the Navy
Washington, DC 20390

1 CDR Richard L. Martin, USN
COMFAIRMIRAMAR F-14
NAS Miramar, CA 92145

1 Research Director, Code 06
Research and Evaluation Department
U.S. Naval Examining Center
Great Lakes, IL 60088
ATTN: C. S. Winiewicz

1 Chief
Bureau of Medicine and Surgery
Code 413
Washington, DC 20372

1 Program Coordinator
Bureau of Medicine and Surgery (Code 71G)
Department of the Navy
Washington, DC 20372

1 Commanding Officer
Naval Medical Neuropsycniatric
Research Unit
San Diego, CA 92152

1 Dr. John J. Collins
Chief of Naval Operations (OP-987F)
Department of the Navy
Washington, DC 20350

1 Technical Library (Pers-11B)
Bureau of Naval Personnel
Department of the Navy
Washington, DC 20360

1 Technical Director
Naval Personnel Research and
Development Center
San Diego, CA 92152

1 Commanding Officer
Naval Personnel Research and
Development Center
San Diego, CA 92152

1 Superintendent
Naval Postgraduate School
Monterey, CA 92940
ATTN: Library (Code 2124)

1 Mr. George N. Graine
Naval Ship Systems Command
(SHIPS 03H)
Department of the Navy
Washington, DC 20360

1 Technical Library
Naval Ship Systems Command
National Center, Building 3
Room 3S08
Washington, DC 20360

1 Commanding Officer
Service School Command
U.S. Naval Training Center
San Diego, CA 92133
ATTN: Code 303

1 Chief of Naval Training Support
Code N-21
Building 45
Naval Air Station
Pensacola, FL 32508

1 Dr. William L. Maloy
Principal Civilian Advisor
for Education and Training
Naval Training Command, Code 01A
Pensacola, FL 32508

1 Mr. Arnold Rubinstein
Naval Material Command (NMAT-03424)
Room 820, Crystal Plaza No. 6
Washington, DC 20360

Army

1 Commandant
U.S. Army Institute of Administration
Fort Benjamin Harrison, IN 46216
ATTN: EA

1 Armed Forces Staff College
Norfolk, VA 23511
ATTN: Library

1 Director of Research
U.S. Army Armor Human Research Unit
Building 2422, Morade Street
Fort Knox, KY 40121
ATTN: Library

1 U.S. Army Research Institute for the
Behavioral and Social Sciences
1300 Wilson Boulevard
Arlington, VA 22209

1 Commanding Officer
USACDC - PASA
Ft. Benjamin Harrison, IN 46249
ATTN: LTC Montgomery

1 Dr. John L. Kobrick
Military Stress Laboratory
U.S. Army Research Institute of
Environmental Medicine
Natick, MA 01760

1 Commandant
United States Army Infantry School
Fort Benning, GA 31905
ATTN: ATSIN-H

1 U.S. Army Research Institute
Commonwealth Building, Room 239
1300 Wilson Boulevard
Arlington, VA 22209
ATTN: Dr. R. Dusek

1 Mr. Edmund F. Fuchs
U.S. Army Research Institute
1300 Wilson Boulevard
Arlington, VA 22209

1 Chief, Unit Training and Educational
Technology Systems
U.S. Army Research Institute for the
Behavioral and Social Sciences
1300 Wilson Boulevard
Arlington, VA 22209

1 Commander
U.S. Theater Army Support Command,
Europe
APO New York 09058
ATTN: Asst. DCSPER (Education)

1 Dr. Stanley L. Cohen
Work Unit Area Leader
Organizational Development Work Unit
Army Research Institute for Behavioral
and Social Science
1300 Wilson Boulevard
Arlington, VA 22209

1 Dr. Leon H. Nawrocki
U.S. Army Research Institute
Rosslyn Commonwealth Building
1300 Wilson Boulevard
Arlington, VA 22209

Air Force

1 Headquarters, U.S. Air Force
Chief, Personnel Research and Analysis
Division (AF/DPSY)
Washington, DC 20330

1 Research and Analysis Division
AF/DPXYR, Room 4C200
Washington, DC 20330

1 AFHRL/AS (Dr. G. A. Eckstrand)
Wright-Patterson AFB
Ohio 45433

1 AFHRL (AST/Dr. Ross L. Morgan)
Wright-Patterson AFB
Ohio 45433

1 AFHRL/MD
701 Prince Street
Room 200
Alexandria, VA 22314

1 AFOSR (NL)
1400 Wilson Boulevard
Arlington, VA 22209

1 Commandant
USAF School of Aerospace Medicine
Aeromedical Library (SUL-4)
Brooks AFB, TX 78235

1 CAPT Jack Thorpe, USAF
Department of Psychology
Bowling Green State University
Bowling Green, OH 43403

1 Headquarters, Electronic Systems Division
LG Hanscom Field
Bedford, MA 01730
ATTN: Dr. Sylvia R. Mayer/MCIT

Marine Corps

1 COL George Caridakis
Director, Office of Manpower
Utilization
Headquarters, Marine Corps (AO1H)
MCB
Quantico, VA 22134

1 Dr. A. L. Slafkosky
Scientific Advisor (Code Ax)
Commandant of the Marine Corps
Washington, DC 20380

1 Mr. E. A. Dover
Manpower Measurement Unit (Code AO1M-2)
Arlington Annex, Room 2413
Arlington, VA 20370

Coast Guard

1 Mr. Joseph J. Cowan, Chief
Psychological Research Branch (P-1)
U.S. Coast Guard Headquarters
400 Seventh Street, SW
Washington, DC 20590

Other DOD

1 Lt. Col. Austin W. Kibler, Director
Human Resources Research Office
Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

1 Dr. Helga Yeich, Director
Program Management, Defense Advanced
Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

1 Mr. William J. Stormer
DOD Computer Institute
Washington Navy Yard
Building 175
Washington, DC 20374

1 Mr. Ralph R. Canter
Director for Manpower Research
Office of Secretary of Defense
The Pentagon, Room 3C980
Washington, DC 20301

Other Government

1 Office of Computer Information
Institute for Computer Sciences
and Technology
National Bureau of Standards
Washington, DC 20234

Miscellaneous

1 Dr. Scarvia Anderson
Executive Director for Special
Development
Educational Testing Service
Princeton, NJ 08540

1 Dr. Richard C. Atkinson
Department of Psychology
Stanford University
Stanford, CA 94305

1 Dr. Bernard M. Bass
Management Research Center
University of Rochester
Rochester, NY 14627

1 Mr. Edmund C. Berkeley
Berkeley Enterprises, Inc.
815 Washington Street
Newtonville, MA 02160

1 Dr. David G. Bowers
University of Michigan
Institute for Social Research
P.O. Box 1248
Ann Arbor, MI 48106

1 Mr. H. Dean Brown
Stanford Research Institute
333 Ravenswood Avenue
Menlo Park, CA 94025

1 Mr. Michael W. Brown
Operations Research, Inc.
1400 Spring Street
Silver Spring, MD 20910

1 Dr. Ronald P. Carver
American Institutes for Research
8555 Sixteenth Street
Silver Spring, MD 20910

1 Century Research Corporation
4113 Lee Highway
Arlington, VA 22207

1 Dr. Kenneth E. Clark
University of Rochester
College of Arts and Sciences
River Campus Station
Rochester, NY 14627

1 Dr. Allan M. Collins
Bolt Beranek and Newman
50 Moulton Street
Cambridge, MA 02138

1 Dr. René V. Dawis
University of Minnesota
Department of Psychology
Minneapolis, MN 55455

2 ERIC
Processing and Reference Facility
4833 Rugby Avenue
Bethesda, MD 20014

1 Dr. Victor Fields
Department of Psychology
Montgomery College
Rockville, MD 20850

1 Dr. Edwin A. Fleishman
American Institutes for Research
8555 Sixteenth Street
Silver Spring, MD 20910

1 Dr. Robert Glaser, Director
Learning Research and Development
Center
University of Pittsburgh
Pittsburgh, PA 15213

1 Dr. Albert S. Glickman
American Institutes for Research
8555 Sixteenth Street
Silver Spring, MD 20910

1 Dr. Duncan N. Hansen
Center for Computer-Assisted
Instruction
Florida State University
Tallahassee, FL 32306

1 Dr. Henry J. Hamburger
University of California
School of Social Sciences
Irvine, CA 92664

1 Dr. Richard S. Hatch
Decision Systems Associates, Inc.
11428 Rockville Pike
Rockville, MD 20852

1 Dr. M. D. Havron
Human Sciences Research, Inc.
Westgate Industrial Park
7710 Old Springhouse Road
McLean, VA 22101

1 Human Resources Research Organization
Division No. 3
P.O. Box 5787
Presidio of Monterey, CA 93940

1 Human Resources Research Organization
Division No. 4, Infantry
P.O. Box 2086
Fort Benning, GA 31905

1 Human Resources Research Organization
Division No. 5, Air Defense
P.O. Box 6057
Fort Bliss, TX 79916

1 Human Resources Research Organization
Division No. 6, Library
P.O. Box 428
Fort Rucker, AL 36360

1 Dr. Lawrence B. Johnson
Lawrence Johnson and Associates, Inc.
200 S Street, N.W., Suite 502
Washington, DC 20009

1 Dr. Norman J. Johnson
School of Urban and Public Affairs
Carnegie-Mellon University
Pittsburgh, PA 15213

1 Dr. David Klahr
Graduate School of Industrial
Administration
Carnegie-Mellon University
Pittsburgh, PA 15213

1 Dr. Robert R. Mackie
Human Factors Research, Inc.
6780 Cortona Drive
Santa Barbara Research Park
Goleta, CA 93017

1 Dr. Andrew R. Molnar
Technological Innovations in Education
National Science Foundation
Washington, DC 20550

1 Dr. Leo Munday, Vice President
American College Testing Program
P.O. Box 168
Iowa City, IA 52240

1 Dr. Donald A. Norman
Center for Human Information Processing
University of California, San Diego
La Jolla, CA 92037

1 Mr. Luigi Petrullo
2431 North Edgewood Street
Arlington, VA 22207

1 Dr. Robert D. Pritchard
Department of Psychology
Purdue University
Lafayette, IN 47907

1 Dr. Diane M. Ramsey-Klee
R-K Research & System Design
3947 Ridgemont Drive
Malibu, CA 90265

1. Dr. Joseph W. Rigney
Behavioral Technology Laboratories
University of Southern California
3717 South Grand
Los Angeles, CA 90007

1 Dr. Leonard L. Rosenbaum, Chairman
Department of Psychology
Montgomery College
Rockville, MD 20850

1 Dr. George E. Rowland
Rowland and Company, Inc.
P.O. Box 61
Haddonfield, NJ 08033

1 Dr. Arthur I. Siegel
Applied Psychological Services
Science Center
404 East Lancaster Avenue
Wayne, PA 19087

1 Mr. Dennis J. Sullivan
725 Benson Way
Thousand Oaks, CA 91360

1 Dr. Benton J. Underwood
Northwestern University
Department of Psychology
Evanston, IL 60201

1 Dr. David J. Weiss
Department of Psychology
University of Minnesota
Minneapolis, MN 55455

1 Dr. Anita West
Denver Research Institute
University of Denver
Denver, CO 80210

1 Dr. Kenneth Wexler
School of Social Sciences
University of California
Irvine, CA 92664

1 Dr. John Annett
The Open University
Milton Keynes
Buckinghamshire, ENGLAND

1 Maj. P. J. DeLeo
Instructional Technology Branch
AF Human Resources Laboratory
Lowry AFB, CO 80230

1 Dr. Martin Rockway
Technical Training Division
Lowry Air Force Base
Denver, CO 80230

1 Dr. Eric McWilliams, Program Manager
Technology and Systems, TIE
National Science Foundation
Washington, DC 20550

1 Dr. Milton S. Katz
MITRE Corporation
Westgate Research Center
McLean, VA 22101

1 Dr. Charles A. Ullmann
Director, Behavioral Sciences Studies
Information Concepts Incorporated
1701 No. Ft. Myer Drive
Arlington, VA 22209

FILMED FROM BEST AVAILABLE COPY

(Continued from inside front cover)

96 R. C. Atkinson, J. W. Brelsford, and R. M. Shiffrin. Multi-process models for memory with applications to a continuous presentation task. April 13, 1966. (J. math. Psychol., 1967, 4, 277-300).

97 P. Suppes and E. Crothers. Some remarks on stimulus-response theories of language learning. June 12, 1966.

98 R. Bjork. All-or-none subprocesses in the learning of complex sequences. (J. math. Psychol., 1968, 1, 182-195).

99 E. Gammon. The statistical determination of linguistic units. July 1, 1966.

100 P. Suppes, L. Hyman, and M. Jerman. Linear structural models for response and latency performance in arithmetic. (In J. P. Hill (ed.), Minnesota Symposia on Child Psychology. Minneapolis, Minn.: 1967. Pp. 160-200).

101 J. L. Young. Effects of intervals between reinforcements and test trials in paired-associate learning. August 1, 1966.

102 H. A. Wilson. An investigation of linguistic unit size in memory processes. August 3, 1966.

103 J. T. Townsend. Choice behavior in a cued-recognition task. August 8, 1966.

104 W. H. Batchelder. A mathematical analysis of multi-level verbal learning. August 9, 1966.

105 H. A. Taylor. The observing response in a cued psychophysical task. August 10, 1966.

106 R. A. Bjork. Learning and short-term retention of paired associates in relation to specific sequences of interpresentation intervals. August 11, 1966.

107 R. C. Atkinson and R. M. Shiffrin. Some Two-process models for memory. September 30, 1966.

108 P. Suppes and C. Ihrke. Accelerated program in elementary-school mathematics--the third year. January 30, 1967.

109 P. Suppes and I. Rosenthal-Hill. Concept formation by kindergarten children in a card-sorting task. February 27, 1967.

110 R. C. Atkinson and R. M. Shiffrin. Human memory: a proposed system and its control processes. March 21, 1967.

111 Theodore S. Rodgers. Linguistic considerations in the design of the Stanford computer-based curriculum in initial reading. June 1, 1967.

112 Jack M. Knutson. Spelling drills using a computer-assisted instructional system. June 30, 1967.

113 R. C. Atkinson. Instruction in initial reading under computer control: the Stanford Project. July 14, 1967.

114 J. W. Brelsford, Jr. and R. C. Atkinson. Recall of paired-associates as a function of overt and covert rehearsal procedures. July 21, 1967.

115 J. H. Stelzer. Some results concerning subjective probability structures with semiorders. August 1, 1967

116 D. E. Rumelhart. The effects of interpresentation intervals on performance in a continuous paired-associate task. August 11, 1967.

117 E. J. Fishman, L. Keller, and R. E. Atkinson. Massed vs. distributed practice in computerized spelling drills. August 18, 1967.

118 G. J. Groen. An investigation of some counting algorithms for simple addition problems. August 21, 1967.

119 H. A. Wilson and R. C. Atkinson. Computer-based instruction in initial reading: a progress report on the Stanford Project. August 25, 1967.

120 F. S. Roberts and P. Suppes. Some problems in the geometry of visual perception. August 31, 1967. (Synthese, 1967, 17, 173-201)

121 D. Jamison. Bayesian decisions under total and partial ignorance. D. Jamison and J. Kozielecki. Subjective probabilities under total uncertainty. September 4, 1967.

122 R. C. Atkinson. Computerized instruction and the learning process. September 15, 1967.

123 W. K. Estes. Outline of a theory of punishment. October 1, 1967.

124 T. S. Rodgers. Measuring vocabulary difficulty: An analysis of item variables in learning Russian-English and Japanese-English vocabulary parts. December 18, 1967.

125 W. K. Estes. Reinforcement in human learning. December 20, 1967.

126 G. L. Wolford, D. L. Wessel, W. K. Estes. Further evidence concerning scanning and sampling assumptions of visual detection models. January 31, 1968.

127 R. C. Atkinson and R. M. Shiffrin. Some speculations on storage and retrieval processes in long-term memory. February 2, 1968.

128 John Holmgren. Visual detection with imperfect recognition. March 29, 1968.

129 Lucille B. Mlodnosky. The Frostig and the Bender Gestalt as predictors of reading achievement. April 12, 1968.

130 P. Suppes. Some theoretical models for mathematics learning. April 15, 1968 (Journal of Research and Development in Education, 1967, 1, 5-22)

131 G. M. Olson. Learning and retention in a continuous recognition task. May 15, 1968.

132 Ruth Norene Hartley. An investigation of list types and cues to facilitate initial reading vocabulary acquisition. May 29, 1968.

133 P. Suppes. Stimulus-response theory of finite automata. June 19, 1968.

134 N. Moler and P. Suppes. Quantifier-free axioms for constructive plane geometry. June 20, 1968. (In J. C. H. Gerretsen and F. Oort (Eds.), Compositio Mathematica. Vol. 20. Groningen, The Netherlands: Wolters-Noordhoff, 1968. Pp. 143-152.)

135 W. K. Estes and D. P. Horst. Latency as a function of number of response alternatives in paired-associate learning. July 1, 1968.

136 M. Schlag-Rey and P. Suppes. High-order dimensions in concept identification. July 2, 1968. (Psychom. Sci., 1968, 11, 141-142)

137 R. M. Shiffrin. Search and retrieval processes in long-term memory. August 15, 1968.

138 R. D. Freund, G. R. Loftus, and R.C. Atkinson. Applications of multiprocess models for memory to continuous recognition tasks. December 18, 1968.

139 R. C. Atkinson. Information delay in human learning. December 18, 1968.

140 R. C. Atkinson, J. E. Holmgren, and J. F. Juola. Processing time as influenced by the number of elements in the visual display. March 14, 1969.

141 P. Suppes, E. F. Loftus, and M. Jerman. Problem-solving on a computer-based teletype. March 25, 1969.

142 P. Suppes and Mona Morningstar. Evaluation of three computer-assisted instruction programs. May 2, 1969.

143 P. Suppes. On the problems of using mathematics in the development of the social sciences. May 12, 1969.

144 Z. Domotor. Probabilistic relational structures and their applications. May 14, 1969.

145 R. C. Atkinson and T. D. Wickens. Human memory and the concept of reinforcement. May 20, 1969.

146 R. J. Titiev. Some model-theoretic results in measurement theory. May 22, 1969.

147 P. Suppes. Measurement: Problems of theory and application. June 12, 1969.

148 P. Suppes and C. Ihrke. Accelerated program in elementary-school mathematics--the fourth year. August 7, 1969.

149 D. Rundus and R.C. Atkinson. Rehearsal in free recall: A procedure for direct observation. August 12, 1969.

150 P. Suppes and S. Feldman. Young children's comprehension of logical connectives. October 15, 1969.

(Continued on back cover)

FILMED FROM BEST AVAILABLE COPY

(Continued from inside back cover)

151 Joaquim H. Laubsch. An adaptive teaching system for optimal item allocation. November 14, 1969.
152 Roberta L. Klatzky and Richard C. Atkinson. Memory scans based on alternative test stimulus representations. November 25, 1969.
153 John E. Holmgren. Response latency as an indicant of information processing in visual search tasks. March 16, 1970.
154 Patrick Suppes. Probabilistic grammars for natural languages. May 15, 1970.
155 E. Gammon. A syntactical analysis of some first-grade readers. June 22, 1970.
156 Kenneth N. Wexler. An automaton analysis of the learning of a miniature system of Japanese. July 24, 1970.
157 R. C. Atkinson and J.A. Paulson. An approach to the psychology of instruction. August 14, 1970.
158 R.C. Atkinson, J.D. Fletcher, H.C. Chetin, and C.M. Stauffer. Instruction in initial reading under computer control: the Stanford project. August 13, 1970.
159 Dewey J. Rundus. An analysis of rehearsal processes in free recall. August 21, 1970.
160 R.L. Klatzky, J.F. Juola, and R.C. Atkinson. Test stimulus representation and experimental context effects in memory scanning.
161 William A. Rottmayer. A formal theory of perception. November 13, 1970.
162 Elizabeth Jane Fishman Loftus. An analysis of the structural variables that determine problem-solving difficulty on a computer-based teletype. December 18, 1970.
163 Joseph A. Van Campen. Towards the automatic generation of programmed foreign-language instructional materials. January 11, 1971.
164 Jamesine Friend and R.C. Atkinson. Computer-assisted instruction in programming: AID. January 25, 1971.
165 Lawrence James Hubert. A formal model for the perceptual processing of geometric configurations. February 19, 1971.
166 J. F. Juola, I.S. Fischler, C.T. Wood, and R.C. Atkinson. Recognition time for information stored in long-term memory.
167 R.L. Klatzky and R.C. Atkinson. Specialization of the cerebral hemispheres in scanning for information in short-term memory.
168 J.D. Fletcher and R.C. Atkinson. An evaluation of the Stanford CAI program in initial reading (grades K through 3). March 12, 1971.
169 James F. Juola and R.C. Atkinson. Memory scanning for words versus categories.
170 Ira S. Fischler and James F. Juola. Effects of repeated tests on recognition time for information in long-term memory.
171 Patrick Suppes. Semantics of context-free fragments of natural languages. March 30, 1971.
172 Jamesine Friend. Instruct coders' manual. May 1, 1971.
173 R.C. Atkinson and R.M. Shiffrin. The control processes of short-term memory. April 19, 1971.
174 Patrick Suppes. Computer-assisted instruction at Stanford. May 19, 1971.
175 D. Jamison, J.D. Fletcher, P. Suppes, and R.C. Atkinson. Cost and performance of computer-assisted instruction for compensatory education.
176 Joseph Offir. Some mathematical models of individual differences in learning and performance. June 28, 1971.
177 Richard C. Atkinson and James F. Juola. Factors influencing speed and accuracy of word recognition. August 12, 1971.
178 P. Suppes, A. Goldberg, G. Kanz, B. Searle, and C. Stauffer. Teacher's handbook for CAI courses. September 1, 1971.
179 Adele Goldberg. A generalized instructional system for elementary mathematical logic. October 11, 1971.
180 Max Jerman. Instruction in problem solving and an analysis of structural variables that contribute to problem-solving difficulty. November 12, 1971.
181 Patrick Suppes. On the grammar and model-theoretic semantics of children's noun phrases. November 29, 1971.
182 Georg Kreisel. Five notes on the application of proof theory to computer science. December 10, 1971.
183 James Michael Moloney. An investigation of college student performance on a logic curriculum in a computer-assisted instruction setting. January 28, 1972.
184 J.E. Friend, J.D. Fletcher, and R.C. Atkinson. Student performance in computer-assisted instruction in programming. May 10, 1972.
185 Robert Lawrence Smith, Jr. The syntax and semantics of ERICA. June 14, 1972.
186 Adele Goldberg and Patrick Suppes. A computer-assisted instruction program for exercises on finding axioms. June 23, 1972.
187 Richard C. Atkinson. Ingredients for a theory of instruction. June 26, 1972.
188 John D. Bonvillian and Veda R. Charrow. Psycholinguistic implications of deafness: A review. July 14, 1972.
189 Phipps Arabie and Scott A. Boorman. Multidimensional scaling of measures of distance between partitions. July 26, 1972.
190 John Ball and Dean Jamison. Computer-assisted instruction for dispersed populations: System cost models. September 15, 1972.
191 W. R. Sanders and J. R. Ball. Logic documentation standard for the Institute for Mathematical Studies in the Social Sciences. October 4, 1972.
192 M.T. Kane. Variability in the proof behavior of college students in a CAI course in logic as a function of problem characteristics. October 6, 1972.
193 P. Suppes. Facts and fantasies of education. October 18, 1972.
194 R. C. Atkinson and J. F. Juola. Search and decision processes in recognition memory. October 27, 1972.
195 P. Suppes, R. Smith , and M. Léveillé. The French syntax and semantics of PHILIPPE, part 1 : Noun phrases. November 3, 1972.
196 D. Jamison, P. Suppes, and S. Wells. The effectiveness of alternative instructional methods: A survey. November 1972.
197 P. Suppes. A survey of cognition in handicapped children. December 29, 1972.
198 B. Searle, P. Lorton, Jr., A. Goldberg, P. Suppes, N. Ledet, and C. Jones. Computer-assisted instruction program: Tennessee State University. February 14, 1973.
199 D. R. Levine. Computer-based analytic grading for German grammar instruction. March 16, 1973.
200 P. Suppes, J.D. Fletcher, M. Zanotti, P. V. Lorton, Jr., and B. W. Searle. Evaluation of computer-assisted instruction in elementary mathematics for hearing-impaired students. March 17, 1973.
201 G. A. Huff. Geometry and formal linguistics. April 27, 1973.
202 C. Jensema. Useful techniques for applying latent trait mental-test theory. May 9, 1973.
203 A. Goldberg. Computer-assisted instruction: The application of theorem-proving to adaptive response analysis. May 25, 1973.
204 R. C. Atkinson, D. J. Herrmann, and K. T. Wescourt. Search processes in recognition memory. June 8, 1973.
205 J. Van Campen. A computer-based introduction to the morphology of Old Church Slavonic. June 18, 1973.
206 R. B. Kimball. Self-optimizing computer-assisted tutoring: Theory and practice. June 25, 1973.
207 R.C. Atkinson, J.D. Fletcher, E. J. Lindsay, J.O. Campbell, and A. Barr. Computer-assisted instruction in initial reading. July 9, 1973.
208 V. R. Charrow and J. D. Fletcher. English as the second language of deaf students. July 20, 1973.
209 J. A. Paulson. An evaluation of instructional strategies in a simple learning situation. July 30, 1973.
210 N. Martin. Convergence properties of a class of probabilistic adaptive schemes called sequential reproductive plans. July 31, 1973.
211 J. Friend. Computer-assisted instruction in programming: A curriculum description. July 31, 1973.